

AD667981

AB

## USAALABS TECHNICAL REPORT 67-61B

### GENERAL METHOD FOR DETERMINING THE AERODYNAMIC CHARACTERISTICS OF FAN-IN-WING CONFIGURATIONS

#### VOLUME II COMPUTER PROGRAM DESCRIPTION

By

G. R. Nink  
R. F. Gilbert  
K. A. Sundstrom

DDC  
DAE2016  
APR 15 1968

December 1967

U. S. ARMY AVIATION MATERIEL LABORATORIES  
FORT EUSTIS, VIRGINIA

CONTRACT DA 44-177-AMC-323(T)

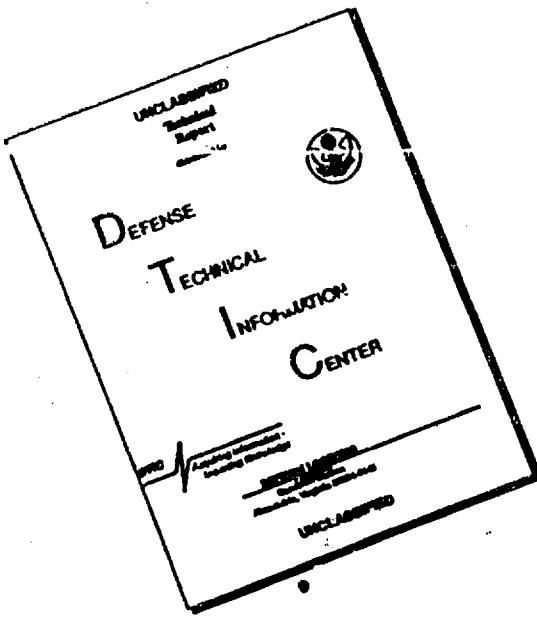
THE BOEING COMPANY  
RENTON, WASHINGTON

This document has been approved  
for public release and sale; its  
distribution is unlimited.



335

# **DISCLAIMER NOTICE**



**THIS DOCUMENT IS BEST  
QUALITY AVAILABLE. THE COPY  
FURNISHED TO DTIC CONTAINED  
A SIGNIFICANT NUMBER OF  
PAGES WHICH DO NOT  
REPRODUCE LEGIBLY.**

### Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission, to manufacture, use, or sell any patented invention that may in any way be related thereto.

### Disposition Instructions

Destroy this report when no longer needed. Do not return it to originator

ACCESSION NO.	
CSSTI	WHITE SECTION <input checked="" type="checkbox"/>
DDC	BWF SECTION <input type="checkbox"/>
UNANNOUNCED	<input type="checkbox"/>
JUSTIFICATION	
DRAFTED BY: APPROVED BY: CIVIL SERVICE CODES	
B.C.L. A.A.L. AND OR SPECIAL	
/	



DEPARTMENT OF THE ARMY  
U. S. ARMY AVIATION MATERIEL LABORATORIES  
FORT EUSTIS, VIRGINIA 23604

This report has been reviewed by the U. S. Army Aviation Materiel Laboratories and is considered to be technically sound.

A study was made of the inlet flow to a fan-in-wing configuration as a function of various flow parameters as influenced by the geometry of the wing and of the fan inlet under various flight conditions. The theoretical results were compared with the available experimental data. The resulting computer program is a powerful tool for use in investigating not only fan-in-wing configurations but arbitrary wings with and without fans and with and without flaps.

The report is published for the dissemination of information and the stimulation of ideas in the application of theoretical aerodynamics.

Task 1F125901A14234  
Contract DA 44-177-AMC-323(T)  
USAALABS Technical Report 67-61B  
December 1967

A GENERAL METHOD FOR DETERMINING THE AERODYNAMIC  
CHARACTERISTICS OF FAN-IN-WING CONFIGURATIONS

Volume II  
Computer Program Description

By

G. R. Hink  
R. F. Gilbert  
K. A. Sundstrom

Prepared by

The Boeing Company  
Renton, Washington

for

U.S. ARMY AVIATION MATERIEL LABORATORIES  
FORT EUSTIS, VIRGINIA

This document has been approved  
for public release and sale; its  
distribution is unlimited.

## SUMMARY

This second volume of the report describes the digital computer program developed to study the aerodynamic characteristics of fan-in-wing configurations. Volume I (Reference 1) of this report provides the details of the aerodynamic theory underlying the computer program.

The method presented in Volume I has proved feasible for computers with sufficient capacity to solve the large number of linear simultaneous equations required for accurate geometric representation of the configuration. The resulting computer program was written in the FORTRAN IV and ASCENT programming languages for the Control Data Corporation 6000-series digital computers. To provide versatility, the computer program is divided into three basic packages called a geometry program, a potential-flow program, and a boundary-layer program. The outputs of the first and second programs are directly usable in the second and third programs, respectively, or there may be supplementary information or interpretation provided between programs.

The geometry program produces a detailed geometric description of the configuration under study. The potential-flow program provides a theoretical solution of the flow about the configuration. The boundary-layer program indicates the relative boundary-layer condition on the inlet geometry.

This volume is also a guide for program maintenance. A general discussion of the three programs is followed by a detailed description of each. The section on program usage contains the computer hardware and software requirements, input data formats, program field length requirements, timing and output estimates, and necessary monitor control cards. Flow charts, segmentation structure diagrams, and detailed descriptions of each subroutine are contained in the appendixes.

TABLE OF CONTENTS

	<u>Page</u>
SUMMARY . . . . .	iii
LIST OF ILLUSTRATIONS . . . . .	vi
LIST OF SYMBOLS . . . . .	xi
1. INTRODUCTION . . . . .	1
2. DESCRIPTION . . . . .	2
2.1 Geometry Program . . . . .	2
2.2 Potential-Flow Program . . . . .	3
2.3 Boundary-Layer Program . . . . .	6
3. PROGRAM USAGE . . . . .	7
3.1 Geometry Program . . . . .	7
3.1.1 Machine Components . . . . .	7
3.1.2 Input Data Format . . . . .	7
3.1.3 Monitor Control Cards . . . . .	39
3.1.4 Program Field Length, Timing, and Output Estimate . . . . .	41
3.2 Potential-Flow Program . . . . .	41
3.2.1 Machine Components . . . . .	41
3.2.2 Input Data Format . . . . .	41
3.2.3 Monitor Control Cards . . . . .	73
3.2.4 Program Field Length, Timing, and Output Estimate . . . . .	76
3.3 Boundary-Layer Program . . . . .	76
3.3.1 Machine Components . . . . .	76
3.3.2 Input Data Format . . . . .	76
3.3.3 Monitor Control Cards . . . . .	80
3.3.4 Program Field Length, Timing, and Output Estimate . . . . .	82
4. REFERENCES . . . . .	83
APPENDIX I—FLOW CHARTS . . . . .	84
APPENDIX II—SEGMENTATION STRUCTURE . . . . .	95
APPENDIX III—SUBROUTINE DESCRIPTIONS . . . . .	97
A. Geometry program subroutine descriptions . . . . .	97
B. Potential-flow program subroutine descriptions . . . . .	153
C. Boundary-layer program subroutine descriptions . . . . .	310
DISTRIBUTION . . . . .	320

## LIST OF ILLUSTRATIONS

<u>Figure</u>		<u>Page</u>
1	Data Card Arrangement for Geometry Program . . . . .	9
2	Data Card Arrangement for Subroutine WING . . . . .	13
3	Data Card Arrangement for Subroutine LIFT . . . . .	18
4	Data Card Arrangement for Subroutine INLET . . . . .	26
5	Data Card Arrangement for Region 1 . . . . .	27
6	Data Card Arrangement for Region 2 . . . . .	28
7	Data Card Arrangement for Region 3 . . . . .	29
8	Data Card Arrangement for Region 4 . . . . .	30
9	Data Card Arrangement for Region 5 . . . . .	31
10	Data Card Arrangement for Region 6 . . . . .	32
11	Data Card Arrangement for Subroutine TUBE . . . . .	35
12	Data Card Arrangement for Subroutine AXISYM . . . . .	38
13	Monitor Control Card Arrangement for Geometry Program . . . . .	40
14	Source and Quadrilateral Vortex Networks . . . . .	44
15	Column Designations . . . . .	44
16	Single Multihorseshoe Vortex Arrangement . . . . .	46
17	Multihorseshoe Vortex Network Designations . . . . .	47
18	Barrier Networks . . . . .	50
19	Initial Streamline Data . . . . .	52
20	Data Card Arrangement for Potential-Flow Program . . . . .	55
21	Data Card Arrangement for Geometric Section . . . . .	57
22	Data Card Arrangement for Source-Panel Geometry . . . . .	59
23	Data Card Arrangement for Quadrilateral Vortex Geometry . . . . .	62

**LIST OF ILLUSTRATIONS (Continued)**

<u>Figure</u>	<u>Page</u>
24 Data Card Arrangement for Multihorseshoe Vortex Geometry . . . . .	66
25 Data Card Arrangement for Off-Body Points . . . . .	67
26 Data Card Arrangement for Aerodynamic Section . . . . .	72
27 Monitor Control Card Arrangement for Potential-Flow Program . . . . .	75
28 Potential-Flow Program Central Processor Time Estimate . . . . .	77
29 Data Card Arrangement for Boundary-Layer Program . . . . .	79
30 Monitor Control Card Arrangement for Boundary-Layer Program	81
31 Fan-In-Wing Problem Flow Chart . . . . .	84
32 Geometry Program Flow Chart . . . . .	85
33 Geometry Program—Subroutine WING (Wing Paneling) . . . . .	86
34 Geometry Program—Subroutine LIFT (Lifting System) . . . . .	87
35 Geometry Program—Subroutine INLET (Inlet Region Paneling) . . . . .	88
36 Geometry Program—Subroutine TUBE (Efflux Tube Paneling) . . . . .	89
37 Potential-Flow Program Flow Chart . . . . .	90
38 Potential-Flow Program—Subroutine GEOM (Preliminary Geometry)	91
39 Potential-Flow Program—Subroutine AERO (Velocity Matrix Formation) . . . . .	92
40 Potential-Flow Program—Subroutine MATRIX (Matrix Equation Solution) . . . . .	92
41 Potential-Flow Program—Subroutine FLOW (Potential-Flow Solution)	93
42 Potential-Flow Program—Subroutine STREAM (Streamline Trajectories) . . . . .	93
43 Boundary-Layer Program Flow Chart . . . . .	94
44 Geometry Program Segmentation Structure . . . . .	95

LIST OF ILLUSTRATIONS (Continued)

<u>Figure</u>		<u>Page</u>
45	Potential-Flow Program Segmentation Structure . . . . .	96
46	Subroutine ACS—Panel Diagonals . . . . .	98
47	Subroutine AXISYM—Polar Coordinate System . . . . .	100
48	Subroutine CROSS—Intersection of Two Functions . . . . .	104
49	Subroutine CURFIT—Airfoil Definition . . . . .	107
50	Subroutine INLET—Region 1 of the Inlet . . . . .	114
51	Subroutine INLET—Regions 2 and 3 of the Inlet . . . . .	114
52	Subroutine INLET—Region 4 of the Inlet . . . . .	115
53	Subroutine INLET—Region 5 of the Inlet . . . . .	115
54	Subroutine INLET—Paneling Region 1 . . . . .	116
55	Subroutine INLET—Paneling Region 2 . . . . .	117
56	Subroutine LIFT—Lifting System Geometry . . . . .	119
57	Subroutine LIFT—Flow at the Trailing Edge . . . . .	119
58	Subroutine OBLIQ—Wing-Cylinder Geometry . . . . .	125
59	Subroutine OBLIQ—Wing-Cylinder Intersection . . . . .	126
60	Subroutine RCOSIN—Lift-Fan Orientation . . . . .	131
61	Subroutine SELECT—Airfoil Coordinate Calculation . . . . .	139
62	Subroutine TUBE—Jet Efflux Tube Geometry . . . . .	146
63	Subroutine TUBE—Tube Paneling . . . . .	147
64	Subroutine WING—Defining Airfoil Section . . . . .	150
65	Subroutine WING—Wing Geometry . . . . .	150
66	Subroutine WING—Wing Paneling . . . . .	150
67	Subroutine AERO—Normal Velocity Matrices . . . . .	156

LIST OF ILLUSTRATIONS (Continued)

<u>Figure</u>	<u>Page</u>
68 Subroutine CONSUR—Lift-Fan Assembly . . . . .	162
69 Subroutine CONSUR—Barrier Singularities . . . . .	163
70 Subroutine FORCES— Lift-Fan Assembly . . . . .	189
71 Subroutine GEOM—Flow Diagram . . . . .	208
72 Subroutine PANEL—Source-Panel Network . . . . .	228
73 Subroutine SFLOW—Velocity Component Matrix . . . . .	248
74 Subroutine SFLOW—Quadrilateral Vortex . . . . .	250
75 Subroutine SFLOW—Multihorseshoe Vortex . . . . .	251
76 Subroutine SFLOW1—Streamline Velocity Component Matrix . .	256
77 Subroutine SFLOW1—Quadrilateral Vortex . . . . .	258
78 Subroutine SFLOW1—Multihorseshoe Vortex . . . . .	258
79 Subroutine SORDQ—Source Panel . . . . .	264
80 Subroutine SORDQ—Triangular Panels . . . . .	264
81 Subroutine SORDQ—Source-Panel Diagonals . . . . .	265
82 Subroutine SORDQ— Source-Panel Centroid . . . . .	268
83 Subroutine SOREQ—Influence of a Source Panel . . . . .	274
84 Subroutine SOREQ—Local Coordinate System . . . . .	274
85 Subroutine UFLOW—Velocity Component Matrix . . . . .	289
86 Subroutine UFLOW—Quadrilateral Vortex . . . . .	290
87 Subroutine UFLOW—Multihorseshoe Vortex . . . . .	291
88 Subroutine UFLOW1—Streamline Velocity Component Matrix . .	295
89 Subroutine UFLOW1—Quadrilateral Vortex . . . . .	296
90 Subroutine UFLOW1—Multihorseshoe Vortex . . . . .	297

LIST OF ILLUSTRATIONS (Concluded)

<u>Figure</u>		<u>Page</u>
91 Subroutine VORDQ—Quadrilateral Vortex . . . . .		301
92 Subroutine VORDQ—Triangular Vortices . . . . .		301
93 Subroutine VOREQ—Influence of a Line Vortex . . . . .		305
94 Subroutine VOREQ—Line Vortex in a Symmetric Flow . . . . .		307
95 Subroutine INTERP—Lagrangian Interpolation Scheme . . . . .		316

### LIST OF SYMBOLS

A	Source-panel area
$A_b$	Barrier area
$A_c$	Centerbody base area
a	Direction cosine
B	Required normal component of velocity
$b_r$	Reference span
$\hat{b}$	Unit vector in plane normal to velocity
$C_D$	Drag coefficient
$C_F$	Force coefficient
$C_f$	Skin friction coefficient
$C_L$	Lift coefficient
$C_m$	Moment coefficient
$C_p$	Pressure coefficient
$C_{p_c}$	Centerbody base pressure coefficient
$C_{p_e}$	Fan exit pressure coefficient
$c_r$	Reference chord
D	Displacement distance of panel corner
$d_c$	Centerbody base diameter
H	Boundary-layer shape factor, $\delta^*/\theta$
h	Distance between barrier and exit planes
$I_\xi, I_{\xi\eta}, I_\eta$	Second moments of inertia of source panel about origin of panel coordinate system
$\hat{n}$	Outward unit normal vector to surface
$\hat{n}_b$	Unit vector directed upward along fan axis

$p$	Static pressure
$R_\infty$	Reynolds number per foot
$r$	Radial distance; also distance between two points
$S_r$	Planform reference area
$s$	Distance along streamline
$t_{\max}$	Longest source-panel diagonal
$\hat{t}$	Unit vector in fan efflux direction
$U_R$	Reference velocity
$U_s$	Specified inflow velocity, $\hat{V} \cdot \hat{n}$
$u, v, w$	Velocity components due to the singularities
$V$	Nondimensional Velocity
$VN$	Normal component of velocity due to a unit strength singularity
$VX, VY, VZ$	Velocity components due to a unit strength singularity
$V_\infty$	Free-stream velocity (assigned the value of unity)
$W$	Weight of a bound multihorseshoe vortex segment
$x, y, z$	Cartesian coordinates
$x_b, y_b, z_b$	Coordinates of the fan axis intersection with barrier plane
$x_r, y_r, z_r$	Origin of the moment axis

#### Greek Symbols

$\alpha$	Angle of attack in degrees
$\beta$	Streamline divergence angle measured along orthogonal trajectory; thrust vector angle
$\delta^*$	Boundary-layer displacement thickness
$\epsilon$	Small distance
$\theta$	Boundary-layer momentum thickness; also angles around fan
$\xi, \eta, \zeta$	Local coordinate system

$\rho$  Density  
 $\sigma$  Singularity strength  
 $\downarrow$  Angle of yaw in degrees

## 1. INTRODUCTION

This part of the report (Volume II) describes the digital computer program developed to study the aerodynamic characteristics of a lift fan-in-wing configuration. Three separate programs were written, each oriented toward a particular phase of the overall problem. In the order of use, they are: the geometry program, the potential-flow program, and the boundary-layer program. The theoretical considerations underlying the method are presented in Volume I (Reference 1).

The geometry program presents, in a form acceptable to the potential-flow program, a geometric description of the configuration under study. This program geometrically defines any of five different types of surfaces: wings, lifting network systems in wings, fan inlets, jet efflux tubes, and axisymmetric bodies. In general, several of these surfaces are combined to define a given configuration. Coordinates of the surfaces are output on cards and used as input for the potential-flow program.

The potential-flow program provides a theoretical solution of the flow about arbitrary fan-in-wing configurations by means of a potential-flow mathematical model consisting of source and vortex singularities distributed on the wing, wake, and jet efflux tube. The singularity representation is composed of a number of constant-strength source sheets covering the wing surface to give thickness effects, internal vortex filaments emanating from the wing trailing edge to provide circulation and to simulate the trailing vortex sheet, and a vortex lattice to represent the boundary between the jet efflux tube and the exterior flow. The problem is solved by satisfying the boundary condition of parallel flow to the surfaces at a finite number of points, resulting in a complex system of linear algebraic equations to be solved. The final results consist of the velocity and pressure distributions, forces, and moments. In addition, the program calculates streamlines on the model, including streamlines emanating from the inlet of the lift fan. Flow characteristics along the streamlines are punched on data cards as input to the boundary-layer program.

The boundary-layer program analyzes the turbulent boundary layer. Certain simplifying assumptions have been made to allow the introduction of two-dimensional axisymmetric flow analysis.

## **2. DESCRIPTION**

An analysis of aerodynamic characteristics of a fan-in-wing configuration is accomplished using three computer programs in the following order:

- Geometry program
- Potential-flow program
- Boundary-layer program

These programs provide a theoretical solution of the flow about arbitrary fan-in-wing configurations by means of a potential-flow mathematical model consisting of source and vortex singularities distributed on the wing, wake, and jet efflux tube. The potential-flow model is geometrically defined by the geometry program and aerodynamically analyzed by the potential-flow program and the boundary-layer program. The geometry program punches the geometric definition on cards suitable for input to the potential-flow program. The potential-flow program, in turn, punches the data necessary for input to the boundary-layer program. User intervention is necessary to prepare the inputs to all programs.

The programs are written in the FORTRAN IV and ASCENT languages for the Control Data Corporation 6600 (131k) digital computer. Control of the computer is monitored by the SCOPE (Simultaneous Control of Program Execution) Operating System. Information concerning the programming languages and the SCOPE Operating System can be found in References 2, 3, and 4. Because the geometry and potential-flow programs exceed the capacity of a single-core load, the segmentation feature of the loader is used, allowing the programs to be subdivided into several parts called segments. The segments are loaded and executed in a specified order to solve a particular problem.

### **2.1 GEOMETRY PROGRAM**

The purpose of the geometry program is to calculate x, y, and z coordinates of the surfaces of several geometrical shapes in a reference coordinate system. These coordinates may be punched on cards that are used as input for the potential-flow program. The various shapes acceptable by the geometry program are the following:

1. A wing
2. A lifting system
3. A fan inlet
4. A jet efflux tube
5. An axisymmetric body

The geometry program consists of a small main program and several subroutines. The main program reads and interprets control cards, calling the appropriate subroutines into core. The subroutines do all of the calculations necessary to find the surface coordinates. The main subroutines in the geometry program are WING, LIFT, INLET, TUBE, and AXISYM. Any one of these may be called into core by a data card containing the subroutine name. Cards such as these are referred to as "control cards," since they control the flow of program execution.

After a control card has been read, the program expects various types of input, depending on the subroutine called into core. For example, when the program encounters a control card containing the word TUBE in columns 1 through 4, it will expect further input to conform to that required by subroutine TUBE. A detailed input description for each subroutine is given in Section 3.1.2, "Input Data Format."

In addition to the five control cards listed above, the geometry program will recognize three others: CARD, CASE, and EXIT. These control cards will not cause loading and execution of a like-named subroutine, but perform individual functions within the main program. They are also described in detail in Section 3.1.2.

## 2.2 POTENTIAL-FLOW PROGRAM

The potential-flow program provides a theoretical solution of the flow about arbitrary fan-in-wing configurations by means of a potential-flow mathematical model consisting of source and vortex singularities distributed on the wing, wake, and jet efflux tube. The program is divided into two sections: Geometric and Aerodynamic. The Geometric Section provides a suitable geometric description of the potential-flow mathematical model of the configuration, and the Aerodynamic Section solves the theoretical formulation of the flow about the model. Execution begins in program MAIN where program variables are initialized. Then execution passes to subroutine CTRL, which controls the flow through the program by interpreting control cards located within the data deck.

Multiple cases, each involving a different fan-in-wing configuration, can be run. When a nonsystem error occurs during the execution of a case, a partial data printout followed by an error message is given, and execution is passed to the next case. Execution time is a direct function of the number of singularities utilized in representing a configuration. For example, a simple wing solution with 500 singularities can be obtained in 10 to 12 minutes, whereas a sophisticated fan-in-wing model with 1200 singularities may require an hour or more.

The potential-flow model is represented by singularities composed of a number of constant-strength source sheets covering the wing surface to give thickness effects, a vortex lattice forming the boundary between the jet efflux tube and the exterior flow, and internal vortex filaments emanating from the wing trailing edge to provide circulation and to produce the trailing vortex sheet.

Each type of singularity has a name: the constant-strength source sheets are called source panels, the vortex lattice singularities are called quadrilateral vortices, and the internal vortices are called multihorseshoe vortices. The geometrical representation of the fan-in-wing configuration consists of singularity networks, where each network is an m-by-n singularity array. In particular, the wing is composed of one or more source-panel networks, the jet efflux tube consists of one or more quadrilateral vortex networks, and the lifting system consists of one or more multihorseshoe vortex networks. By definition, the multihorseshoe networks are always m-by-1 arrays.

Two types of free-stream flow conditions are permitted: symmetric and unsymmetric. The flow is symmetric when the angle of yaw of the configuration is zero. For symmetric flow, only the right half of the configuration is input; the left half is accounted for by the program.

The primary function of the Geometric Section of the program is to create two data files containing the geometric quantities and the specified normal velocities of the singularities. The first file, called the defining quantities data file, contains geometric parameters describing the type and the orientation of the singularities. The second file, called the boundary point quantities data file, contains the boundary point orientation, the specified normal velocities of the singularities, and the coordinates of specified off-body points.

The Aerodynamic Section of the program has six functions:

1. Formation of the normal velocity matrix equation
2. Calculation of the singularity strengths by solution of a matrix equation
3. Calculation of the velocity and pressure at points on the wing, lift-fan barrier, jet efflux tube, and off the body
4. Calculation of the forces and moments acting on the body
5. Tracing of the streamlines on the body
6. Preparation of streamline flow characteristics for the boundary-layer program

The primary function of the Aerodynamic Section is the calculation of the singularity strengths, which requires the solution of the matrix equation

$$\begin{bmatrix} VN_{ij} \end{bmatrix} \begin{bmatrix} \sigma_{jk} \end{bmatrix} = \begin{bmatrix} B_{ik} \end{bmatrix} \quad (1)$$

where  $VN_{ij}$  = normal velocity at the  $i^{\text{th}}$  boundary point induced by the  $j^{\text{th}}$  unit strength singularity

$\sigma_{jk}$  = strength of the  $j^{\text{th}}$  singularity that satisfies the  $k^{\text{th}}$  solution

$B_{ik}$  = required normal velocity at the  $i^{th}$  boundary point for the  $k^{th}$  solution, which is the sum of the free-stream normal velocity component and the specified normal velocity

The boundary condition of parallel flow to the surface at a finite number of points is used to establish the matrix equation. Several solutions may be requested. A solution is specified by either an angle of attack and angle of yaw, or a zero free-stream velocity. By rearrangement, the matrix equation to be solved is given by

$$\begin{bmatrix} \sigma_{jk} \end{bmatrix} = \begin{bmatrix} VN_{ij} \end{bmatrix}^{-1} \begin{bmatrix} B_{ik} \end{bmatrix} \quad (2)$$

This equation must be solved in partitioned form because of the limited core size of the computer. A package of matrix subroutines developed by The Boeing Company is used to solve the equation. The normal velocity matrix  $\begin{bmatrix} VN_{ij} \end{bmatrix}$  is formed from the velocity component matrices

$$\begin{bmatrix} VN_{ij} \end{bmatrix} = \begin{bmatrix} VX_{ij} \end{bmatrix} \begin{bmatrix} n_{x_i} \end{bmatrix} + \begin{bmatrix} VY_{ij} \end{bmatrix} \begin{bmatrix} n_{y_i} \end{bmatrix} + \begin{bmatrix} VZ_{ij} \end{bmatrix} \begin{bmatrix} n_{z_i} \end{bmatrix} \quad (3)$$

where  $VX_{ij}$

$VY_{ij}$

$VZ_{ij}$

$n_{x_i}$

$n_{y_i}$

$n_{z_i}$

= velocity components at the  $i^{th}$  boundary point induced by the  $j^{th}$  unit strength singularity

= components of the unit normal vector at the  $i^{th}$  boundary point

The elements of the  $\begin{bmatrix} VX_{ij} \end{bmatrix}$ ,  $\begin{bmatrix} VY_{ij} \end{bmatrix}$ ,  $\begin{bmatrix} VZ_{ij} \end{bmatrix}$ ,  $\begin{bmatrix} VN_{ij} \end{bmatrix}$ , and  $\begin{bmatrix} B_{ik} \end{bmatrix}$  matrices are calculated using the geometric quantities in the two data files created by the Geometric Section of the program.

The velocity components at the singularity boundary points on the model and at the off-body points are computed by

$$\begin{aligned} V_{x_{ik}} &= V_{\infty x_k} + \sum_j VX_{ij} \cdot \sigma_{jk} \\ V_{y_{ik}} &= V_{\infty y_k} + \sum_j VY_{ij} \cdot \sigma_{jk} \\ V_{z_{ik}} &= V_{\infty z_k} + \sum_j VZ_{ij} \cdot \sigma_{jk} \end{aligned} \quad (4)$$

where  $V_{\infty x_k}$   
 $V_{\infty y_k}$   
 $V_{\infty z_k}$ } = components of the free-stream velocity for the  $k^{\text{th}}$  solution

The velocity and pressure distributions are calculated by

$$V_{ik} = \sqrt{V_{x_{ik}}^2 + V_{y_{ik}}^2 + V_{z_{ik}}^2} \quad (5)$$

and

$$C_{p_{ik}} = 1 - V_{ik}^2 \quad (6)$$

The velocity and pressure distributions on the fan surface are computed by the method described in subroutine CONSUR (Appendix III). After the velocity and pressure distributions have been computed, the forces and moments on the configuration are calculated. Finally, the streamlines are traced over the configuration, and the flow characteristics along the streamlines are punched on cards for input to the boundary-layer program.

Upon completion, the Aerodynamic Section returns to the Geometric Section to process another configuration.

### 2.3 BOUNDARY-LAYER PROGRAM

This program provides a method of analyzing turbulent boundary layers in three-dimensional flow. Certain simplifying assumptions have been made to allow the introduction of two-dimensional axisymmetric flow analysis.

The method requires the solution of coupled first-order ordinary differential equations that describe the boundary layer. The equations for momentum and moment-of-momentum are solved simultaneously by the Adams-Moulton predictor-corrector method with variable step size. The output from the program includes the skin friction coefficient, momentum thickness, and shape factor at points along the streamline.

### 3. PROGRAM USAGE

A description of program usage for the three programs—geometry program, potential-flow program, and boundary-layer program—is presented in this section. Computer hardware and software requirements, input data formats, program field lengths, and estimates of timing and output are discussed.

#### 3.1 GEOMETRY PROGRAM

##### 3.1.1 Machine Components

The geometry program is coded in FORTRAN IV for the CDC 6600 (131k) digital computer. Control of the computer is monitored by the SCOPE Operating System. Data files for input, printed output, and punched output are required by the program.

##### 3.1.2 Input Data Format

This section describes the use of various capabilities of the geometry program. Complete instructions for submitting geometric problems to the computer are provided.

This program provides a simple method of obtaining x, y, and z coordinates of several types of surfaces on punched cards. The program was written to support the potential-flow program; thus output is oriented toward that program. The types of surfaces that can be produced in a reference coordinate system are:

- A wing
- A lifting system for a wing
- A fan-in-wing inlet
- A jet efflux tube
- Axisymmetric or pseudoaxisymmetric surfaces

The versatile curve-fitting procedure in subroutine WING makes it easy to tailor the chordwise source panel arrangement independently of the airfoil definition. The curve-fitting feature may also enhance the subroutine's usefulness for other applications where accurate intermediate surface coordinates are needed. Subroutine LIFT is geared closely to the potential-flow program both in input and output. The INLET subroutine is for the most part tied closely to fan-in-wing configurations, but the wing contour capability with the inlet makes it potentially useful for other applications where surface intersections are desired. The TUBE subroutine provides the network coordinates for an efflux tube including the influence of velocity ratio, fan diameter, thrust vector angle, and angles of attack and yaw. It will adjust the network arrangement so as to fair smoothly to a nonplanar wing lower surface. Subroutine AXISYM will handle any body that can be paneled in radial planes. Taken together, these subroutines provide a powerful capability to prepare surface coordinates for a wide variety of surfaces.

All geometry program data, except title cards and control cards, are punched in number fields ten columns wide, with six fields per card. Decimal points should always be punched for every input number, and, since their omission is not flagged as an error, users must check their data cards carefully before submitting the run.

Input to the geometry program falls into three categories: control cards, title cards, and numeric input. As their name implies, the control cards control the execution of the program. Eight words are used on control cards:

- |         |           |
|---------|-----------|
| 1) CASE | 5) INLET  |
| 2) CARD | 6) TUBE   |
| 3) WING | 7) AXISYM |
| 4) LIFT | 8) EXIT   |

These control cards must be punched beginning in card column 1 and must not contain any blanks.

The general card-stacking arrangement is shown in Figure 1. If titles are not wanted on the printed output, delete the CASE control card and the two title cards. There are no restrictions as to sequence of the five major geometry subroutines. The functions of the control cards are given below.

Control card CASE: This control card causes execution of two functions. First, an option indicating whether or not card output is desired is set to the no-output condition. Second, the program will read the following two cards as title cards, printing columns 1-80 in several places on the output. If the CASE card is omitted, no title cards are read, and the title for the previous case is used (blank if this is the first case). The card output option remains as for the previous case (no output if this is the first case), unless this option is reset by a CARD control card.

Control card CARD: This card is used when punched-card output is desired. If one of these cards is encountered, card output will be produced for every system (WING, LIFT, INLET, TUBE, AXISYM) following until this option is negated by a CASE control card. A CARD control card is necessary to obtain card output for any system after a CASE control card, regardless of the number of CARD control cards previously used.

The standard format built into the program for card output is the following:

(F10.5, F10.5, F10.5, F10.5, F10.5, F10.5)

Each of the six F10.5's gives the specifications for the corresponding number punched on the output cards. That is, the first F10.5 controls the format of the first number (columns 1-10) on the card; the second F10.5 controls the second number (columns 11-20) on the card; and so on. Under the F10.5 format, all numbers to be punched on cards must be less than 1,000. Numbers too large for the specified format are rendered useless.

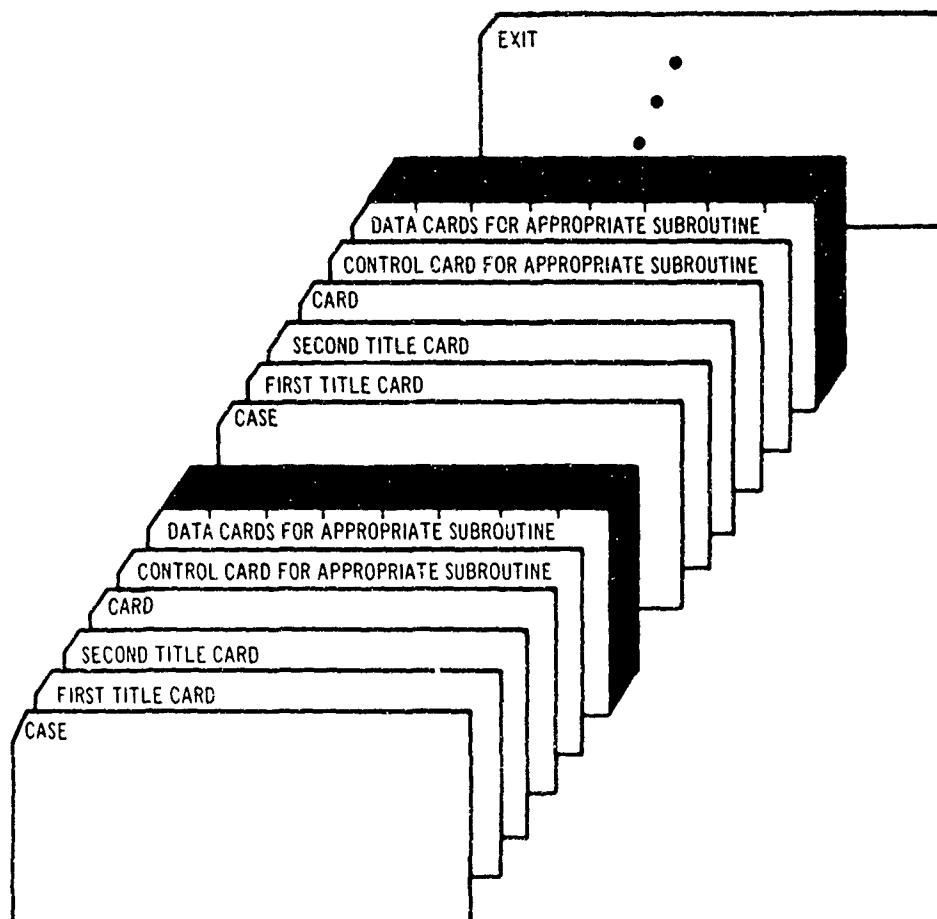


Figure 1. Data Card Arrangement for Geometry Program.

An exception is the LIFT subroutine, where the format has been altered to give numbers up to 10,000 in the first and fourth fields. The modified format is:

(F10.4, F10.5, F10.5, F10.4, F10.5, F10.5)

This alteration was necessary to allow the large x values associated with the trailing edge. The above limitations are built into the program, but they can be overridden if necessary; however, unless such action is taken, these restrictions will be adhered to. It is the responsibility of the user to check that the magnitude of his data is not greater than that allowed.

If a nonstandard format must be used, it is punched in columns 11 through 47 of the pertinent CARD control card in exactly the format displayed above (including parentheses and commas, and no blank spaces). All of the output data to which this CARD control card applies will be punched on cards with the nonstandard format. The allowable range for numbers punched under various formats is tabulated below.

Format	Allowable Range
F10.2	a < 1,000,000.
F10.3	a < 100,000.
F10.4	a < 10,000.
F10.5	a < 1,000.
F10.6	a < 100.
F10.7	a < 10.

Note: a is the expected output number.

Control cards WING, LIFT, INLET, TUBE, and AXISYM: These control cards transfer control to the subroutines for the appropriate system.

Control card EXIT: This control card is the last card of the data deck. When this control card is encountered, program execution is terminated.

**GEOMETRY PROGRAM CARD INPUT**

WING input. — Figure 2 displays the data card arrangement for subroutine WING. A description of the card input to subroutine WING follows.

	<u>Column</u>	<u>Code</u>	<u>Explanation</u>
<u>Card 1</u>	1-4	WING	Control card—contains the word WING.
<u>Card 2</u>	1-10	PARTS	= number of PARTS in this wing. Cards 3-12 must be input PARTS times.
<u>Card 3</u>	1-10 11-20 21-30 31-40 41-50 51-60	XLE <sub>I</sub> YLE <sub>I</sub> ZLE <sub>I</sub> XTE <sub>I</sub> YTE <sub>I</sub> ZTE <sub>I</sub>	coordinates of the leading and trailing edges of the inboard defining section for this wing PART
<u>Card 4</u>	1-10 11-20 21-30 31-40 41-50 51-60	XLE <sub>O</sub> YLE <sub>O</sub> ZLE <sub>O</sub> XLE <sub>O</sub> YLE <sub>O</sub> ZLE <sub>O</sub>	coordinates of the leading and trailing edges of the outward defining section for this wing PART
<u>Card 5</u>	1-10	ORD <sub>I</sub>	= number of pairs of (x/c, z/c) coordinates defining the inboard airfoil section = 0.; coordinates will be used from the outward section of the previous wing PART.  Note: ORD <sub>I</sub> may not be zero for the first PART, as there is no previous PART from which to obtain coordinates.
			$0. \leq ORD_I \leq 200.$
<u>Card Set 6</u>	1-10 11-20 21-30 31-40 41-50 51-60	(x/c) <sub>1</sub> (z/c) <sub>1</sub> (x/c) <sub>2</sub> (z/c) <sub>2</sub> (x/c) <sub>3</sub> (z/c) <sub>3</sub>	coordinates defining the inboard airfoil section. There must be ORD <sub>I</sub> of these pairs, six numbers per card. If ORD <sub>I</sub> is zero, delete this card set.
<u>Card 7</u>	1-10	ORD <sub>O</sub>	= number of pairs of (x/c, z/c) coordinates defining the outward airfoil section

<u>Column</u>	<u>Code</u>	<u>Explanation</u>
		= 0.; coordinates will be used from inboard section of this wing PART.
		$0. \leq ORD_O \leq 200.$
<u>Card Set 8</u>	1-10 $(x/c)_1$ 11-20 $(z/c)_1$ 21-30 $(x/c)_2$ 31-40 $(z/c)_2$ 41-50 $(x/c)_3$ 51-60 $(z/c)_3$	coordinates defining the outboard airfoil section. There must be $ORD_O$ of these pairs, six numbers per card. If $ORD_O$ is zero, delete this card set.
<u>Card 9</u>	1-10      COL	= number of spanwise panel divisions in this wing PART  = 0.; the program will output an $M \times 2$ network of panel corner points consisting of the inboard and outboard panel corner points, scaled and given the correct disposition on the planform.  $0. \leq COL \leq 20.$
<u>Card Set 10</u>	1-10 $y_1$ 11-20 $y_2$ 21-30 $y_3$ 31-40 $y_4$ 41-50 $y_5$ 51-60 $y_6$	spanwise panel divisions, input in order of increasing $y_k$ value. There must be COL of these $y_k$ values, six numbers per card. If COL is zero, delete this card set.
<u>Card 11</u>	1-10      ROW	= number of chordwise panel divisions in this wing PART. ROW $(x/c)$ 's are input in card set 12, and a curve-fitting procedure is employed to locate the spanwise generators that form panel edges. In this case, $ORD_I$ need not equal $ORD_O$ .  = 0.; the program uses the inboard and outboard defining coordinates as spanwise generators to form panel edges. Consequently, $ORD_I$ must equal $ORD_O$ . The zero option for $ORD_I$ and $ORD_O$ may still be used for input.  $0. \leq ROW \leq 200.$

	<u>Column</u>	<u>Code</u>	<u>Explanation</u>
<u>Card Set 12</u>	1-10	$(x/c)_1$	x/c values of the chordwise panel divisions.
	11-20	$(x/c)_2$	There must be ROW of these values, six
	21-30	$(x/c)_3$	numbers per card. If ROW is zero, delete
	31-40	$(x/c)_4$	this card set.
	41-50	$(x/c)_5$	
	51-60	$(x/c)_6$	

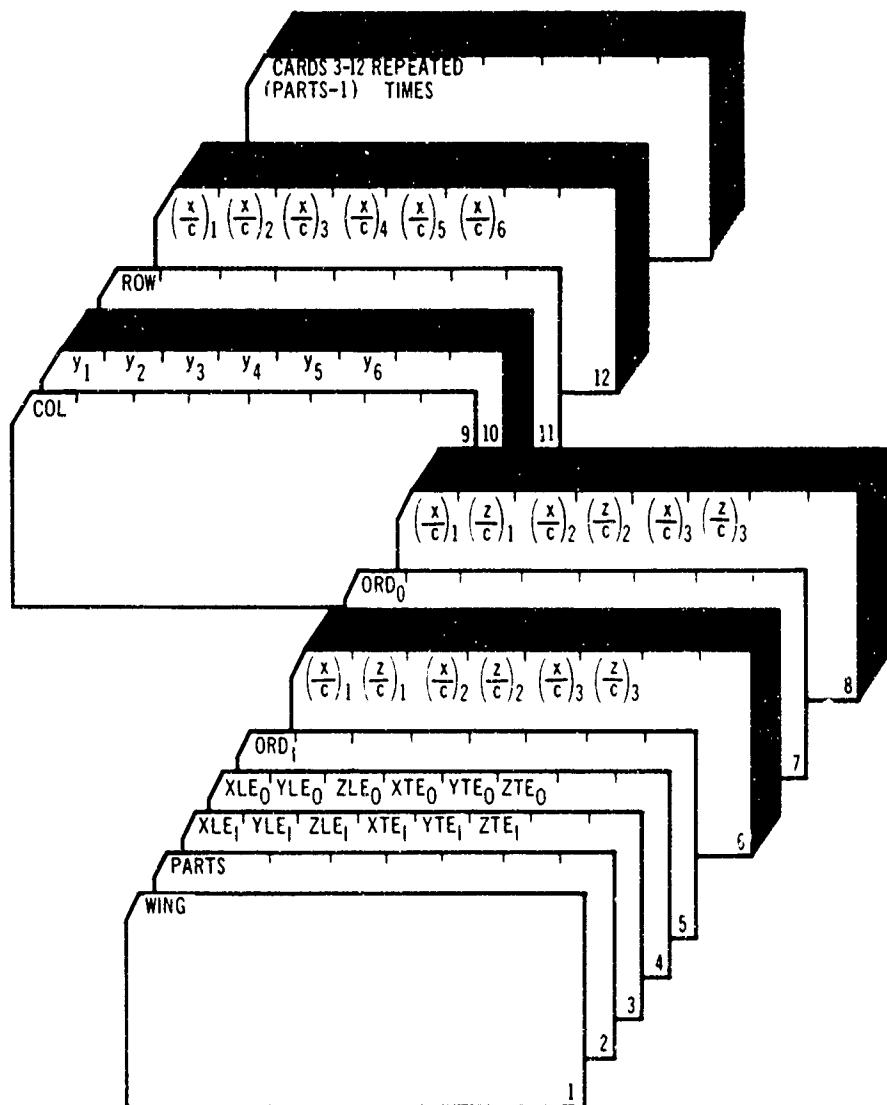


Figure 2. Data Card Arrangement for Subroutine WING.

LIFT input. — Figure 3 displays the data card arrangement for subroutine LIFT.  
The description of the card input to subroutine LIFT follows.

	<u>Column</u>	<u>Code</u>	<u>Explanation</u>
<u>Card 1</u>	1-4	LIFT	Control card—contains the word LIFT.
<u>Card 2</u>	1-10	PARTS	= number of PARTS in this lifting system. Cards 3-21 must be input PARTS times.
<u>Card 3</u>	1-10 11-20 21-30 31-40 41-50 51-60	XLE <sub>I</sub> YLE <sub>I</sub> ZLE <sub>I</sub> XTE <sub>I</sub> YTE <sub>I</sub> ZTE <sub>I</sub>	coordinates of the leading and trailing edges, respectively, of the inboard defining section for this lifting system PART
<u>Card 4</u>	1-10 11-20 21-30 31-40 41-50 51-60	XLE <sub>O</sub> YLE <sub>O</sub> ZLE <sub>O</sub> XTE <sub>O</sub> YTE <sub>O</sub> ZTE <sub>O</sub>	coordinates of the leading and trailing edges, respectively, of the outboard defining section for this lifting system PART
<u>Card 5</u>	1-10	ROWI <sub>I</sub>	<p>= number of (x/c, z/c) coordinate pairs defining the inboard location of the internal vortex corner points (including the one at the trailing edge) at the inboard defining station</p> <p>= 0.; coordinates will be used from the outboard section of the previous lifting system PART.</p> <p>Note: ROWI<sub>I</sub> may not be zero for the first PART, as there is no previous PART from which to obtain coordinates.</p> <p>0. ≤ ROWI<sub>I</sub> ≤ 50.</p>
<u>Card Set 6</u>	1-10 11-20 21-30 31-40 41-50 51-60	(x/c) <sub>1</sub> (z/c) <sub>1</sub> (x/c) <sub>2</sub> (z/c) <sub>2</sub> (x/c) <sub>3</sub> (z/c) <sub>3</sub>	coordinates of the internal vortex corner points at the inboard defining station. There must be ROWI <sub>I</sub> of these pairs, six numbers per card. If ROWI <sub>I</sub> is zero, delete this card set.

	<u>Column</u>	<u>Code</u>	<u>Explanation</u>
<u>Card 7</u>	1-10	ROWI <sub>O</sub>	= number of (x/c, z/c) coordinate pairs defining the internal vortex corner points at the outboard defining station = 0.; coordinates will be used from the inboard section of this lifting system PART. $0. \leq \text{ROWI}_O \leq 50.$
<u>Card Set 8</u>	1-10 11-20 21-30 31-40 41-50 51-60	(x/c) <sub>1</sub> (z/c) <sub>1</sub> (x/c) <sub>2</sub> (z/c) <sub>2</sub> (x/c) <sub>3</sub> (z/c) <sub>3</sub>	coordinates of the internal vortex corner points at the outboard defining station. There must be ROWI <sub>O</sub> of these pairs, six numbers per card. If ROWI <sub>O</sub> is zero, delete this card set.
<u>Card 9</u>	1-10	COL	= number of spanwise divisions (see card set 10) $1. \leq \text{COL} \leq 20.$
<u>Card Set 10</u>	1-10 11-20 21-30 31-40 41-50 51-60	y <sub>1</sub> y <sub>2</sub> y <sub>3</sub> y <sub>4</sub> y <sub>5</sub> y <sub>6</sub>	spanwise location of the vortex segments, input in order of increasing y <sub>k</sub> value. There must be COL of these y <sub>k</sub> values, six numbers per card.
<u>Card 11</u>	1-10	ROWT <sub>I</sub>	= number of trailing segments in the inboard trailing vortex indicating the number of lengths L <sub>i</sub> and direction cosines (a <sub>xj</sub> , a <sub>yz</sub> , a <sub>zi</sub> ) that follow on card sets 12 and 13 for the inboard section = 0.; lengths and direction cosines are to be used from the outboard section of the previous lifting system PART. Note: ROWT <sub>I</sub> may not be zero for the first PART, as there is no previous PART from which to obtain data. $0. \leq \text{ROWT}_I \leq 50.$

	<u>Column</u>	<u>Code</u>	<u>Explanation</u>
<u>Card Set 12</u>	1-10 11-20 21-30 31-40 41-50 51-60	$L_1$ $L_2$ $L_3$ $L_4$ $L_5$ $L_6$	length of each trailing vortex segment for the inboard section. There must be $ROWT_I$ of these lengths, six numbers per card. If $ROWT_I$ is zero, delete this card set.
<u>Card Set 13</u>	1-10 11-20 21-30 31-40 41-50 51-60	$a_{x1}$ $a_{y1}$ $a_{z1}$ $a_{x2}$ $a_{y2}$ $a_{z2}$	direction cosines of each trailing vortex segment for the inboard section. There must be $ROWT_I$ of these three numbers, six numbers per card. If $ROWT_I$ is zero, delete this card set.
<u>Card 14</u>	1-10	$ROWT_O$	= number of trailing segments in the outboard trailing vortex, indicating the number of lengths $L_i$ and direction cosines ( $a_{xi}$ , $a_{yi}$ , $a_{zi}$ ) that follow on card sets 15 and 16 for the outboard section  = 0.; lengths and direction cosines are to be used from the inboard section of this lifting system PART.  $0 \leq ROWT_O \leq 50$ .
<u>Card Set 15</u>	1-10 11-20 21-30 31-40 41-50 51-60	$L_1$ $L_2$ $L_3$ $L_4$ $L_5$ $L_6$	length of each trailing vortex segment for the outboard section. There must be $ROWT_O$ of these lengths, six numbers per card. If $ROWT_O$ is zero, delete this card set.
<u>Card Set 16</u>	1-10 11-20 21-30 31-40 41-50 51-60	$a_{x1}$ $a_{y1}$ $a_{z1}$ $a_{x2}$ $a_{y2}$ $a_{z2}$	direction cosines of each trailing vortex segment for the outboard section. There must be $ROWT_O$ of these three numbers, six numbers per card. If $ROWT_O$ is zero, delete this card set.
<u>Card 17</u>	1-10 11-20 21-30 31-40 41-50 51-60	$(x/c)_1$ $(z/c)_1$ $(x/c)_2$ $(z/c)_2$ $(x/c)_3$ $(z/c)_3$	Cards 17, 18, and 19 contain eight coordinates used in the computation of the trailing-edge boundary point and the direction cosines of the normal to the boundary point. The first four coordinates are the inboard upper and lower surface trailing-edge panel-edge locations. The last four coordinates are for the outboard section.

	<u>Column</u>	<u>Code</u>	<u>Explanation</u>
<u>Card 18</u>	1-10 11-20 21-30 31-40 41-50 51-60	$(x/c)_4$ $(z/c)_4$ $(x/c)_5$ $(z/c)_5$ $(x/c)_6$ $(z/c)_6$	See card 17 explanation.
<u>Card 19</u>	1-10 11-20 21-30 31-40	$(x/c)_7$ $(z/c)_7$ $(x/c)_8$ $(z/c)_8$	See card 17 explanation.
<u>Card 20</u>	1-10 11-20 21-30 21-30	NY YCODE $\epsilon_{BP}$	= number of y values to follow on card set 21 1. $\leq NY \leq 20$ . = 0.; the y values on card set 21 are for the panel edges. = 1.; the y values are for the actual boundary point locations. = 0.; the standard position of the boundary point aft of the trailing edge is desired. The standard offset spacing is 0.1 times the upper surface panel width. = actual value of $\epsilon_{BP}$ in terms of a fraction of the upper panel width, if the nonstandard offset spacing is desired
<u>Card Set 21</u>	1-10 11-20 21-30 31-40 41-50 51-60	$y_1$ $y_2$ $y_3$ $y_4$ $y_5$ $y_6$	spanwise location of the panel edges or boundary point locations, depending on YCODE. There must be NY of these y values, six numbers per card.

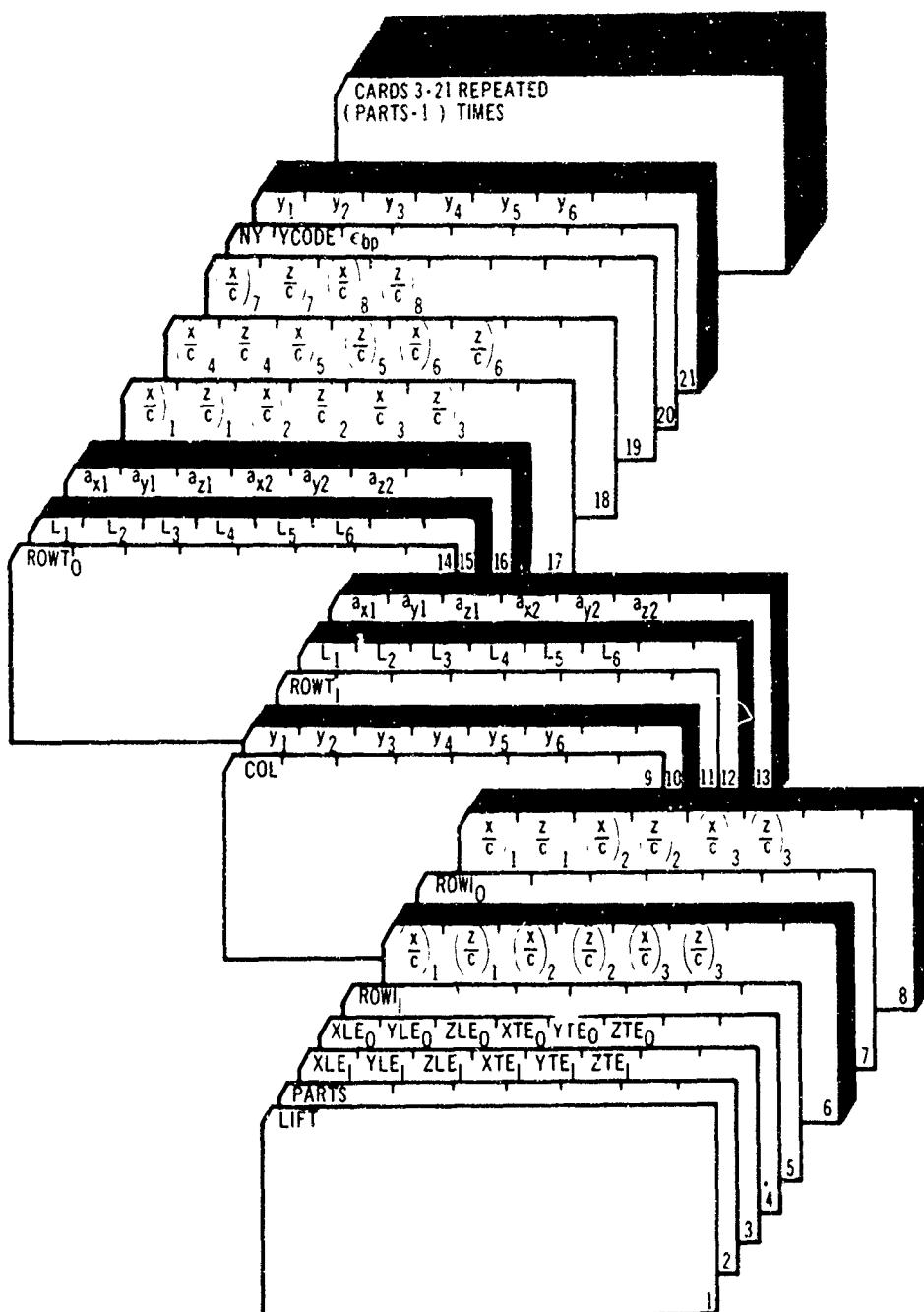


Figure 3. Data Card Arrangement for Subroutine LIFT.

INLET input.—Figures 4 through 10 display the data card arrangement for subroutine INLET. The description of the card input to subroutine INLET follows.

	<u>Column</u>	<u>Code</u>	<u>Explanation</u>
<u>Card 1</u>	1-5	INLET	Control card—contains the word INLET.
<u>Card 2</u>	1-10 11-20 21-30 31-40 41-50 51-60	XLE <sub>I</sub> YLE <sub>I</sub> ZLE <sub>I</sub> XTE <sub>I</sub> YTE <sub>I</sub> ZTE <sub>I</sub>	coordinates of the leading and trailing edge, respectively, of the inboard defining wing section
<u>Card 3</u>	1-10 11-20 21-30 31-40 41-50 51-60	XLE <sub>O</sub> YLE <sub>O</sub> ZLE <sub>O</sub> XTE <sub>O</sub> YTE <sub>O</sub> ZTE <sub>O</sub>	coordinates of the leading and trailing edge, respectively, of the outboard defining wing section
<u>Card 4</u>	1-10	ORD <sub>I</sub>	= number of pairs of (x/c, z/c) coordinates defining the inboard airfoil section  1. $0 \leq ORD_I \leq 200$ .
<u>Card Set 5</u>	1-10 11-20 21-30 31-40 41-50 51-60	(x/c) <sub>1</sub> (z/c) <sub>1</sub> (x/c) <sub>2</sub> (z/c) <sub>2</sub> (x/c) <sub>3</sub> (z/c) <sub>3</sub>	coordinates defining the inboard airfoil section. There must be ORD <sub>I</sub> of these pairs, six numbers per card.
<u>Card 6</u>	1-10	ORD <sub>O</sub>	= number of pairs (x/c, z/c) coordinates defining the outboard airfoil section  = 0.; coordinates will be used from the inboard airfoil section.  0. $0 \leq ORD_O \leq 200$ .
<u>Card Set 7</u>	1-10 11-20 21-30 31-40 41-50 51-60	(x/c) <sub>1</sub> (z/c) <sub>1</sub> (x/c) <sub>2</sub> (z/c) <sub>2</sub> (x/c) <sub>3</sub> (z/c) <sub>3</sub>	coordinates defining the outboard airfoil section. There must be ORD <sub>O</sub> of these pairs, six numbers per card. If ORD <sub>O</sub> is zero, delete this card set.

	<u>Column</u>	<u>Code</u>	<u>Explanation</u>
<u>Card 8</u>	1-10	TYPE	= 0.; wing is defined by lines connecting ordered points. $ORD_O$ must equal $ORD_I$ or be zero. = 1.; wing is defined by lines of constant x/c. $ORD_O$ need not equal $ORD_I$ .
		$\epsilon_T$	tolerance used in the iteration scheme to compute points on the wing surface. A recommended value is $10^{-6}$ times the average wing chord.
<u>Card 9</u>	1-10	$x_o$	$(x_o, y_o, z_o)$ is the origin of the inlet coordinate system on the fan axis. $(n_{xb}, n_{yb}, n_{zb})$ are the direction cosines of the fan axis directed upward.
	11-20	$y_o$	
	21-30	$z_o$	
	31-40	$n_{xb}$	
	41-50	$n_{yb}$	
	51-60	$n_{zb}$	
<u>Card 10.1</u>	1-10	REGION	= 1.; causes program to perform calculations in REGION 1 of the inlet. Cards 11.1-17.1 that follow are for REGION 1.
<u>Card 11.1</u>	1-10	PARTS	= number of PARTS in REGION 1 of the inlet. Cards 12.1-17.1 must be input PARTS times.
<u>Card 12.1</u>	1-10	CODE	= 1.; upper surface points only are computed. = -1.; lower surface points only are computed.
<u>Card 13.1</u>	1-10	M	= number of $F_i$ fraction values to follow on card set 14.1 $1. \leq M \leq 20.$
<u>Card Set 14.1</u>	1-10	$F_1$	fraction values that control panel spacing between two input $(x, y)$ points on the wing planform, input in the order of increasing value.
	11-20	$F_2$	
	21-30	$F_3$	
	31-40	$F_4$	
	41-50	$F_5$	
	51-60	$F_6$	
<u>Card 15.1</u>	1-10	$N_P$	= number of $(x, y)$ pairs to follow on card set 16.1 $1. \leq N_P \leq 200.$

	<u>Column</u>	<u>Code</u>	<u>Explanation</u>
<u>Card Set 16.1</u>	1-10 11-20 21-30 31-40 41-50 51-60	$x_1$ $y_1$ $x_2$ $y_2$ $x_3$ $y_3$	coordinates along the outer boundary of REGION 1; input counterclockwise, viewed from above, for the upper surface and clockwise, viewed from above, for the lower surface. There must be $N_p$ of these (x,y) pairs, six numbers per card.
<u>Card Set 17.1</u>	1-10 11-20 21-30 31-40 41-50 51-60	$x_1$ $y_1$ $x_2$ $y_2$ $x_3$ $y_3$	coordinates along the inner boundary of REGION 1; input counterclockwise, viewed from above, for the upper surface and clockwise, viewed from above, for the lower surface. There must be $N_p$ of these (x,y) pairs, six numbers per card.
<u>Card 10.2</u>	1-10	REGION	= 2.; causes program to perform calculations in REGION 2 of the inlet. Cards 11.2-17.2 that follow are for REGION 2.
<u>Card 11.2</u>	1-10	PARTS	= number of PARTS in REGION 2 of the inlet. Cards 12.2-17.2 must be input PARTS times.
<u>Card 12.2</u>	1-10 11-20	$\theta_{D1}$ $\theta_{D2}$	two defining stations for this inlet PART; input in degrees, counterclockwise when viewed from above  $\theta_{D1} \leq \theta_{D2}$
<u>Card 13.2</u>	1-10	M	= number of (R, Z) pairs to follow on card sets 14.2 and 15.2  $1. \leq M \leq 200.$
<u>Card Set 14.2</u>	1-10 11-20 21-30 31-40 41-50 51-60	$R_1$ $Z_1$ $R_2$ $Z_2$ $R_3$ $Z_3$	coordinates defining the panel corner points along the inlet contour for the <u>first</u> defining station $\theta_{D1}$ ; input beginning on the unmodified wing surface and proceeding down into the inlet. There must be M of these (R, Z) pairs, six numbers per card.
<u>Card Set 15.2</u>	1-10 11-20 21-30 31-40 41-50 51-60	$R_1$ $Z_1$ $R_2$ $Z_2$ $R_3$ $Z_3$	coordinates defining the panel corner points along the inlet contour for the <u>second</u> defining station $\theta_{D2}$ ; input beginning on the unmodified wing surface and proceeding down into the inlet. There must be M of these (R, Z) pairs, six numbers per card.

	<u>Column</u>	<u>Code</u>	<u>Explanation</u>
<u>Card 16.2</u>	1-10	$N_p$	= number of $\theta_p$ paneling stations to follow in card set 17.2  $1 \leq N_p \leq 20.$
<u>Card Set 17.2</u>	1-10 11-20 21-30 31-40 41-50 51-60	$\theta_{P1}$ $\theta_{P2}$ $\theta_{P3}$ $\theta_{P4}$ $\theta_{P5}$ $\theta_{P6}$	$\theta_p$ paneling stations in degrees; input counterclockwise viewed from above. There must be $N_p$ of these $\theta_p$ stations, six numbers per card.  $\theta_{D1} \leq \theta_p \leq \theta_{D2}$
<u>Card 10.3</u>	1-10	REGION	= 3.; causes program to perform calculations in REGION 3 of the inlet. Cards 11.3-18.3 that follow are for REGION 3.
<u>Card 11.3</u>	1-10	PARTS	= number of PARTS in REGION 3 of the inlet. Cards 12.3-18.3 must be input PARTS times.
<u>Card 12.3</u>	1-10	$R_{fan}$	fan radius used to construct panel corner points in REGION 3 of the inlet
<u>Card 13.3</u>	1-10 11-20	$\theta_{D1}$ $\theta_{D2}$	two defining stations for this inlet PART; input in degrees, counterclockwise when viewed from above  $\theta_{D1} < \theta_{D2}$
<u>Card 14.3</u>	1-10	M	= number of Z coordinates to follow on card sets 15.3 and 16.3  $1 \leq M \leq 200.$
<u>Card Set 15.3</u>	1-10 11-20 21-30 31-40 41-50 51-60	$Z_1$ $Z_2$ $Z_3$ $Z_4$ $Z_5$ $Z_6$	coordinates defining the panel corner points along the inlet throat for the first defining station $\theta_{D1}$ ; input beginning at $Z_{ref}$ and proceeding downward to the wing lower surface $Z_b$ . There must be M coordinates, six numbers per card.
<u>Card Set 16.3</u>	1-10 11-20 21-30 31-40 41-50 51-60	$Z_1$ $Z_2$ $Z_3$ $Z_4$ $Z_5$ $Z_6$	coordinates defining the panel corner points along the throat for the second defining station $\theta_{D2}$ ; input beginning at $Z_{ref}$ and proceeding downward to the wing lower surface $Z_b$ . There must be M coordinates, six numbers per card.

	<u>Column</u>	<u>Code</u>	<u>Explanation</u>
<u>Card 17.3</u>	1-10	$N_p$	= number of $\theta_p$ paneling stations to follow on card set 18.3  1. $N_p \leq 20$ .
<u>Card Set 18.3</u>	1-10	$\theta_{P1}$	$\theta_p$ paneling stations in degrees; input counterclockwise viewed from above.
	11-20	$\theta_{P2}$	There must be $N_p$ of these $\theta_p$ stations, six numbers per card.
	21-30	$\theta_{P3}$	
	31-40	$\theta_{P4}$	
	41-50	$\theta_{P5}$	
	51-60	$\theta_{P6}$	$\theta_{D1} \leq \theta_p \leq \theta_{D2}$
<u>Card 10.4</u>	1-10	REGION	= 4.; causes program to perform calculations in REGION 4 of the inlet. Cards 11.4-17.4 that follow are for REGION 4.
<u>Card 11.4</u>	1-10	PARTS	= number of PARTS in REGION 4 of the inlet. Cards 12.4-17.4 must be input PARTS times.
<u>Card 12.4</u>	1-10	$\theta_{D1}$	two defining stations for this inlet PART;
	11-20	$\theta_{D2}$	input in degrees, counterclockwise when viewed from above  $\theta_{D1} \leq \theta_{D2}$
<u>Card 13.4</u>	1-10	M	= number of R coordinates to follow on card sets 14.4 and 15.4  1. $M \leq 200$ .
<u>Card Set 14.4</u>	1-10	$R_1$	radius values defining the panel corner points on the lower surface for the <u>first</u> defining station $\theta_{D1}$ ; input beginning at the edge of the fan exit and proceeding outward.
	11-20	$R_2$	There must be M values, six numbers per card.
	21-30	$R_3$	
	31-40	$R_4$	
	41-50	$R_5$	
	51-60	$R_6$	
<u>Card Set 15.4</u>	1-10	$R_1$	radius values defining the panel corner points on the lower surface for the <u>second</u> defining station $\theta_{D2}$ ; input beginning at the edge of the fan exit and proceeding outward.
	11-20	$R_2$	There must be M values, six numbers per card.
	21-30	$R_3$	
	31-40	$R_4$	
	41-50	$R_5$	
	51-60	$R_6$	
<u>Card 16.4</u>	1-10	$N_p$	= number of $\theta_p$ paneling stations to follow on card set 17.4  1. $N_p \leq 20$ .

	<u>Column</u>	<u>Code</u>	<u>Explanation</u>
<u>Card Set 17.4</u>	1-10 11-20 21-30 31-40 41-50 51-60	$\theta_{P1}$ $\theta_{P2}$ $\theta_{P3}$ $\theta_{P4}$ $\theta_{P5}$ $\theta_{P6}$	$\theta_P$ paneling stations in degrees, input counterclockwise viewed from above. There must be $N_p$ of these $\theta_P$ stations, six numbers per card.
<u>Card 10.5</u>	1-10	REGION	= 5.; causes program to perform calculations in REGION 5 of the inlet. Cards 11.5-17.5 that follow are for REGION 5.
<u>Card 11.5</u>	1-10	PARTS	= number of PARTS in REGION 5 of the inlet. Cards 12.5-17.5 must be input PARTS times.
<u>Card 12.5</u>	1-10 11-20	$\theta_{D1}$ $\theta_{D2}$	two defining stations for this inlet PART; input in degrees, counterclockwise when viewed from above $\theta_{D1} \leq \theta_{D2}$
<u>Card 13.5</u>	1-10	M	= number of (R, Z) pairs to follow on card sets 14.5 and 15.5 1. $\leq M \leq 200$ .
<u>Card 14.5</u>	1-10 11-20 21-30 31-40 41-50 51-60	$R_1$ $Z_1$ $R_2$ $Z_2$ $R_3$ $Z_3$	coordinates defining the interior corner points for the <u>first</u> defining station $\theta_{D1}$ . There must be M of these (R, Z) pairs, six numbers per card.
<u>Card Set 15.5</u>	1-10 11-20 21-30 31-40 41-50 51-60	$R_1$ $Z_1$ $R_2$ $Z_2$ $R_3$ $Z_3$	coordinates defining the interior corner points for the <u>second</u> defining station $\theta_{D2}$ . There must be M of these (R, Z) pairs, six numbers per card.
<u>Card 16.5</u>	1-10	$N_p$	= number of $\theta_P$ paneling stations to follow on card set 17.5 1. $\leq N_p \leq 20$ .

	<u>Column</u>	<u>Code</u>	<u>Explanation</u>
<u>Card Set 17.5</u>	1-10 11-20 21-30 31-40 41-50 51-60	$\theta_{P1}$ $\theta_{P2}$ $\theta_{P3}$ $\theta_{P4}$ $\theta_{P5}$ $\theta_{P6}$	$\theta_P$ paneling stations in degrees, input counterclockwise viewed from above. There must be $N_p$ of these $\theta_P$ stations, six numbers per card.
<u>Card 10.6</u>	1-10	REGION	= 6.; causes program to perform calculations in REGION 6 of the inlet that produces wing surface contours. Cards 11.6-13.6 that follow are for REGION 6.
<u>Card 11.6</u>	1-10	M	= number of points to be computed on the wing surface in each $\theta_P$ profile station. These points are computed on both the upper and lower surfaces and are evenly distributed between RMIN and RMAX.
	11-20	RMIN	minimum radius for which surface points are computed. RMIN may be zero or a negative number, as long as the magnitude does not locate a point off the planform.
	21-30	RMAX	maximum radius for which surface points are computed
<u>Card 12.6</u>	1-10	$N_p$	= number of $\theta_P$ profile stations to follow on card set 13.6  $1 \leq N_p \leq 20.$
<u>Card Set 13.6</u>	1-10 11-20 21-30 31-40 41-50 51-60	$\theta_{P1}$ $\theta_{P2}$ $\theta_{P3}$ $\theta_{P4}$ $\theta_{P5}$ $\theta_{P6}$	$\theta_P$ profile stations in degrees. There must be $N_p$ of these $\theta_P$ stations, six numbers per card.  $0 \leq \theta_P \leq 360.$
<u>Blank Card</u>			blank card used as the last card in the INLET data, causing the program to exit subroutine INLET and to return to the main program

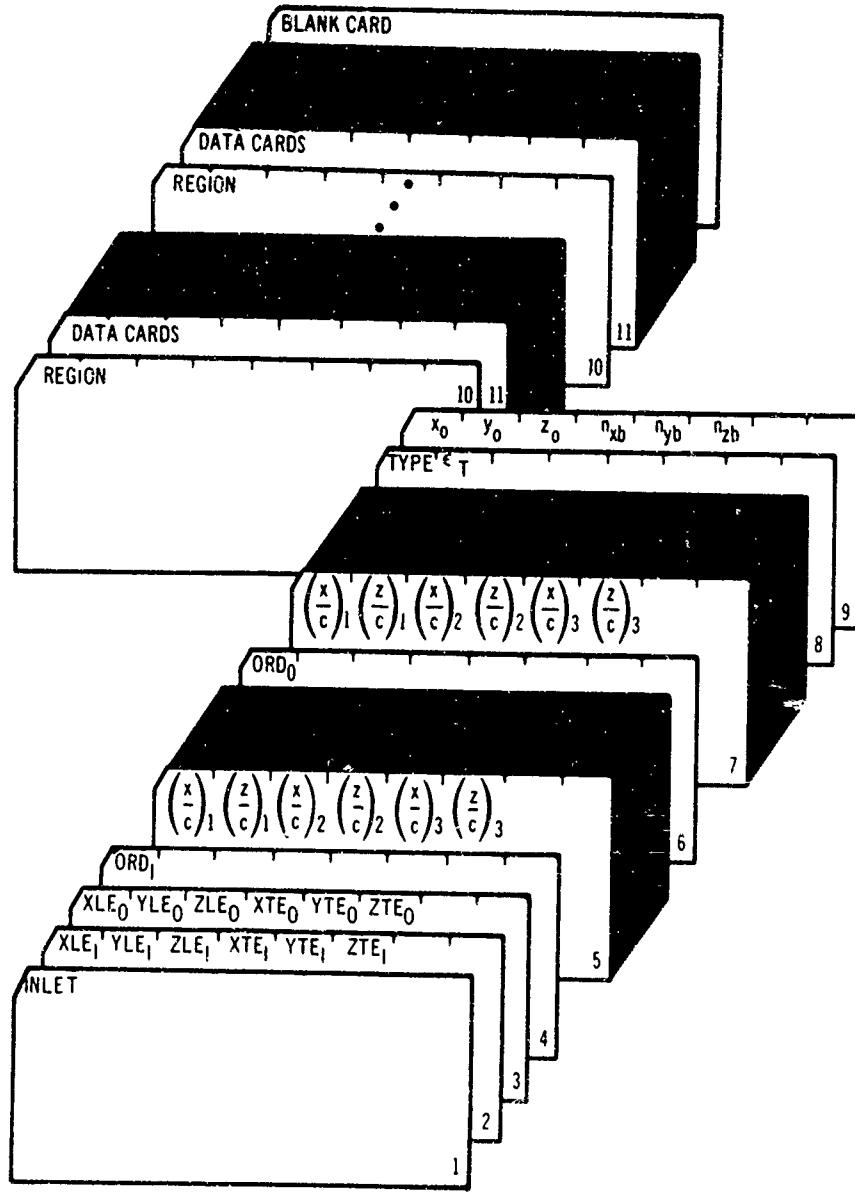


Figure 4. Data Card Arrangement for Subroutine INLET.

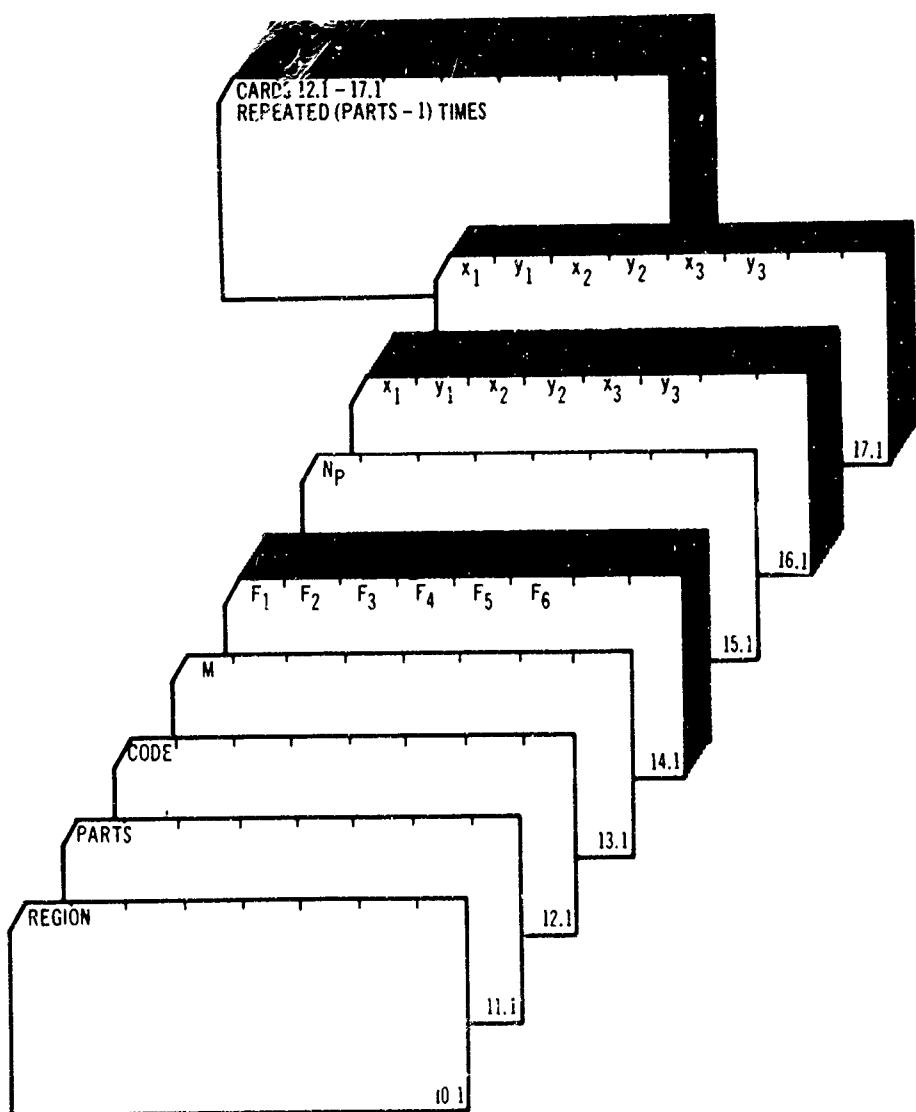


Figure 5. Data Card Arrangement for REGION 1.

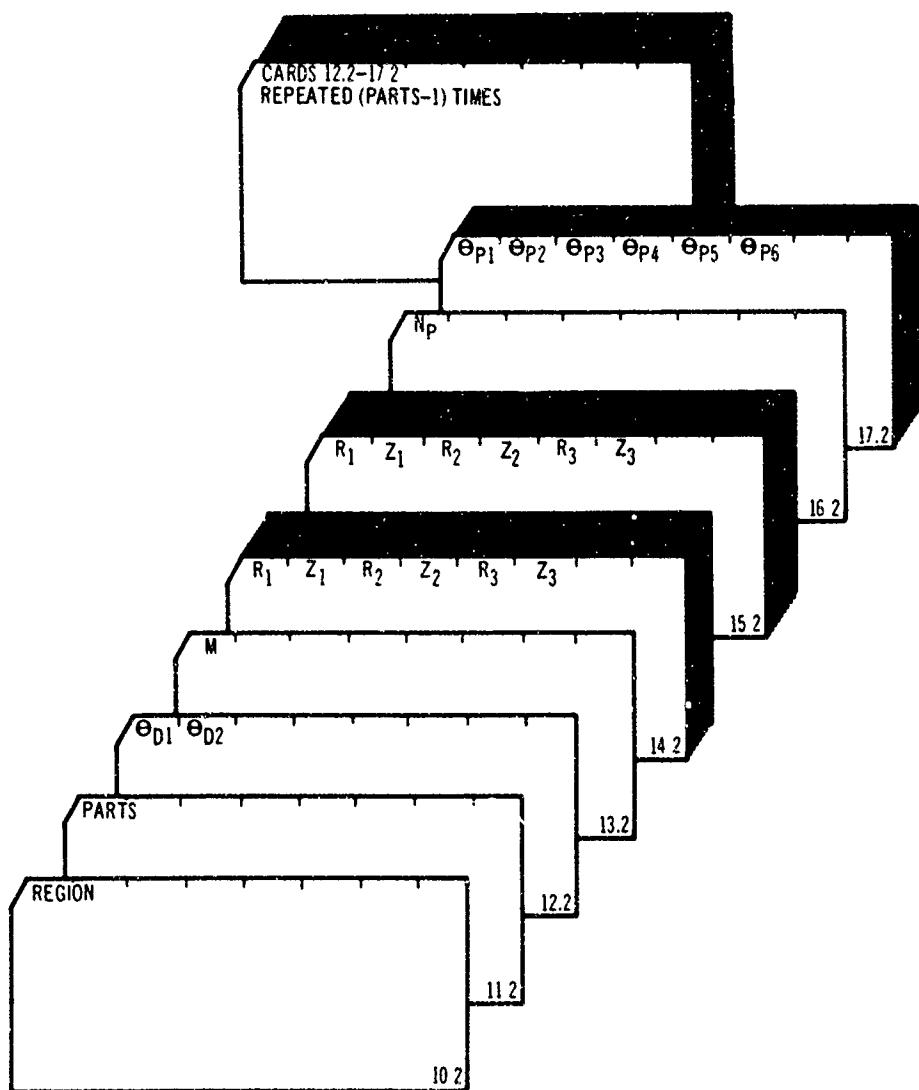


Figure 6. Data Card Arrangement for REGION 2.

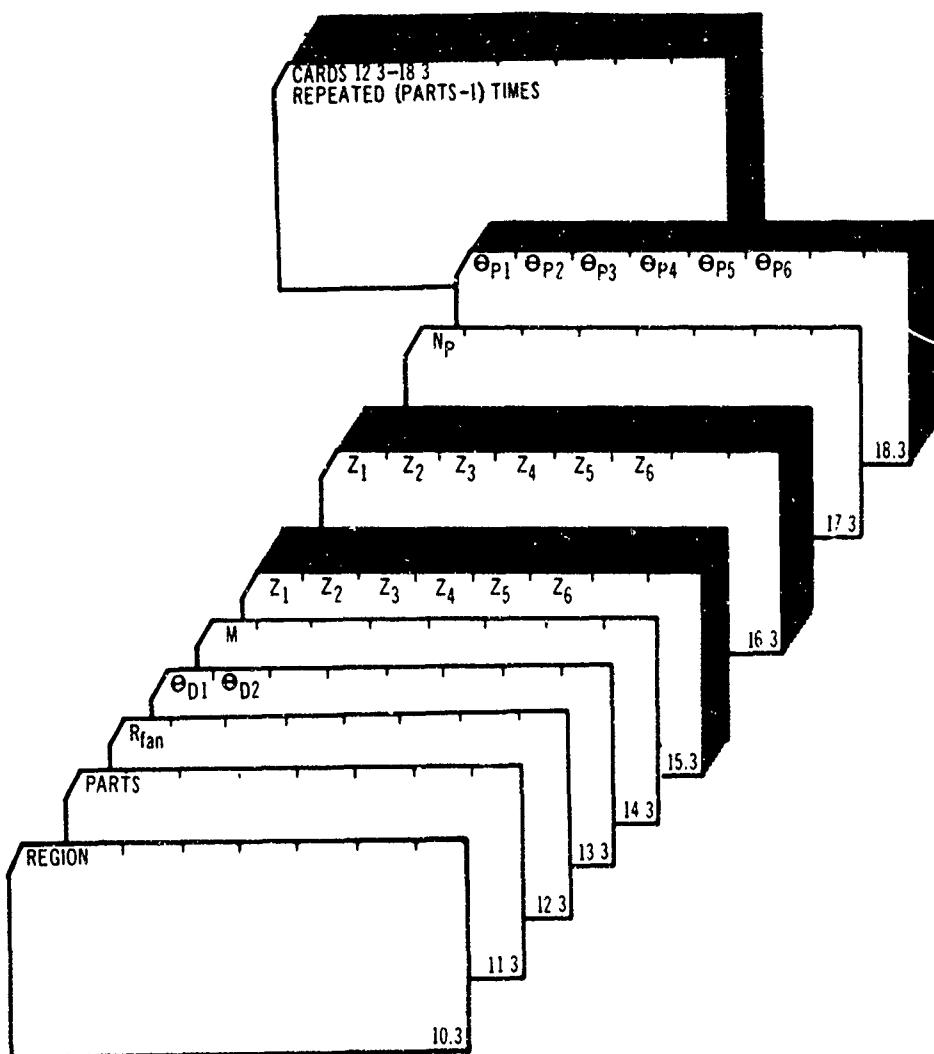


Figure 7. Data Card Arrangement for REGION 3.

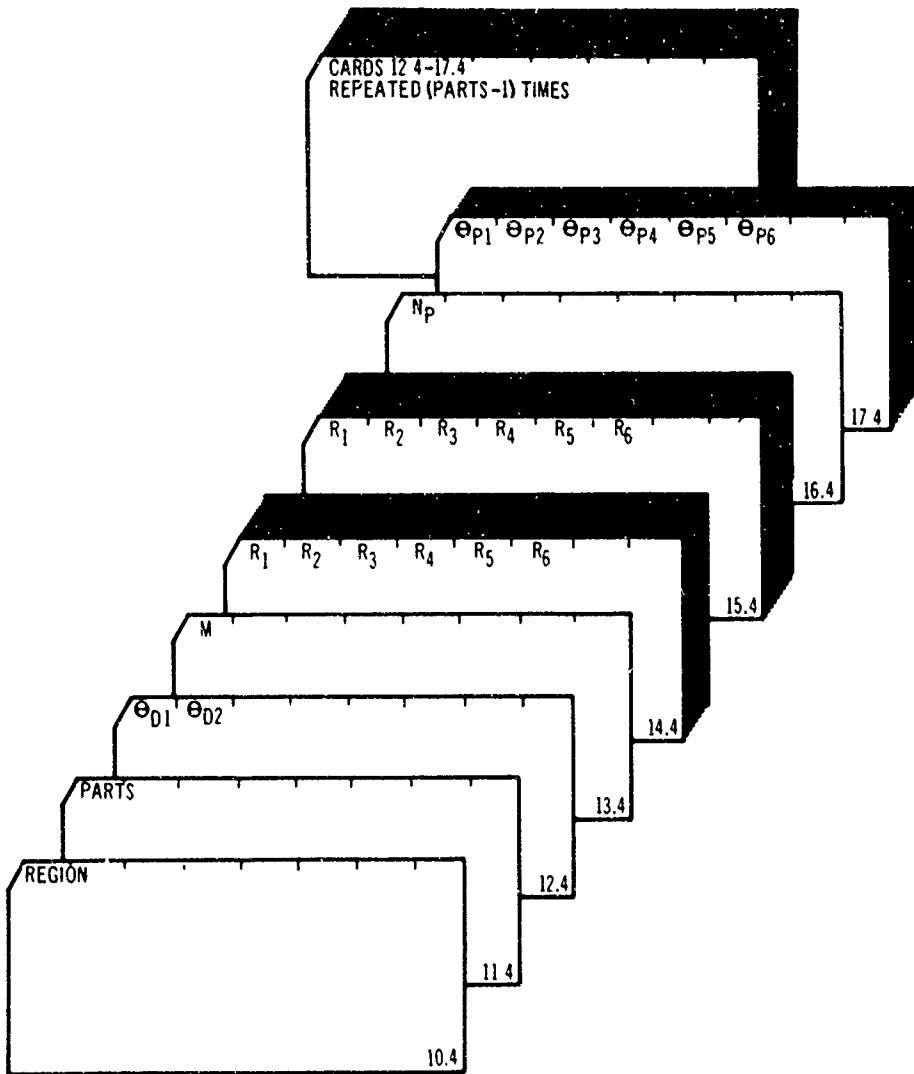


Figure 8. Data Card Arrangement for REGION 4.

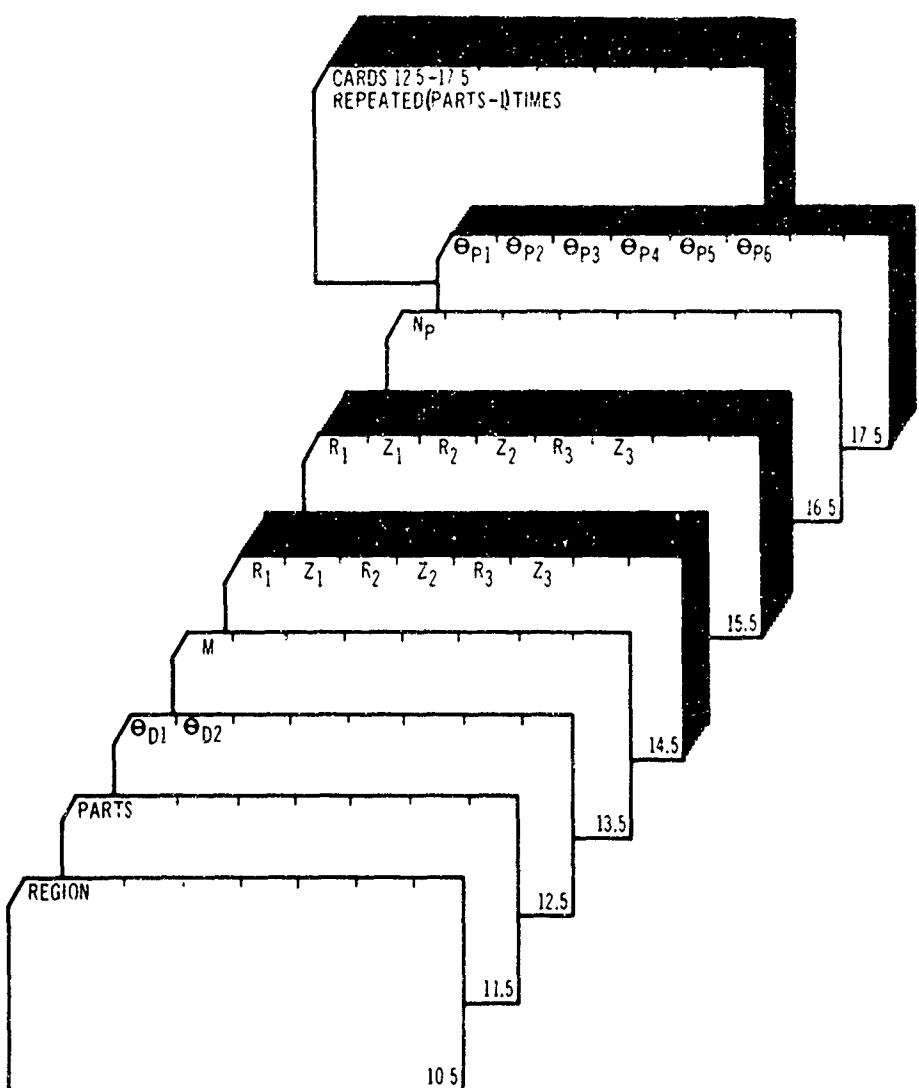


Figure 9. Data Card Arrangement for REGION 5.

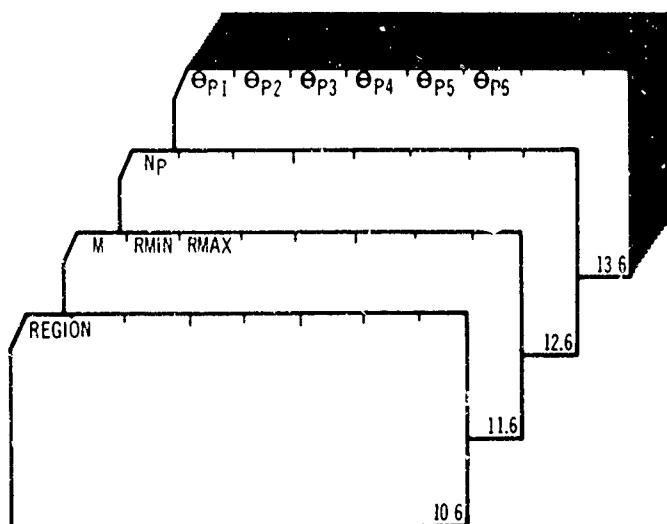


Figure 16. Data Card Arrangement for REGION 6.

TUBE input.—Figure 11 displays the data card arrangement for subroutine TUBE. The description of the card input to subroutine TUBE follows.

	<u>Column</u>	<u>Code</u>	<u>Explanation</u>
<u>Card 1</u>	1-4	TUBE	Control card—contains the word TUBE.
<u>Card 2</u>	1-10 11-20 21-30 31-40 41-50 51-60	$x_o$ $y_o$ $z_o$ $n_o$ $n_{xb}$ $n_{yb}$ $n_{zb}$	( $x_o, y_o, z_o$ ) is a point on the fan axis and ( $n_{xb}, n_{yb}, n_{zb}$ ) are the direction cosines of the fan axis directed upward.
<u>Card 3</u>	1-10 11-20 21-30	$t_x$ $t_y$ $t_z$	direction cosines of the initial jet efflux direction
<u>Card 4</u>	1-10 11-20 21-30 31-40	$V_\infty/V_j$ $\alpha$ $\psi$ $D_{fan}$	ratio of free stream to jet velocity angle of attack, in degrees angle of yaw, in degrees fan exit diameter
<u>Card 5</u>	1-10	M	= number of initial points on the wing lower surface to follow on card set 6  1. $\leq M \leq 50$ .
<u>Card Set 6</u>	1-10 11-20 21-30 31-40 41-50 51-60	$x_1$ $y_1$ $z_1$ $x_2$ $y_2$ $z_2$	coordinates of the initial points on the wing lower surface to which the tube is attached; input counterclockwise viewed from above beginning at any convenient point. If a plane through the fan axis is used as a plane of symmetry for the geometry and flow, only half of the tube is input. For the latter case, the first and last points coincide. There must be M of these initial points, six coordinates per card.
<u>Card 7</u>	1-10	N	= number of arc lengths to follow on card set 8  1. $\leq N \leq 50$ .
<u>Card Set 8</u>	1-10 11-20 21-30 31-40 41-50 51-60	$s_1$ $s_2$ $s_3$ $s_4$ $s_5$ $s_6$	arc lengths measured positive downstream along the jet core trajectory. These define the vortex spacing down the tube.

<u>Column</u>	<u>Code</u>	<u>Explanation</u>
<u>Card 9</u>	1-10 $N_{TR}$	= number of $s_k$ arc lengths over which the tube geometry is to be adjusted to the shape of the wing lower surface
	11-20 $\delta$	small number ( $-1 << \delta << 1$ ) defining the maximum deviation of the vortex spacing on opposite sides of the tube axis and controlling the rate at which cross sections become perpendicular to the tube axis
	21-30 $\epsilon_{bp}$	small positive number (0.005-0.01) controlling the distance of the boundary point below the wing surface. $\epsilon_{bp}$ is the ratio of the distance from the wing surface to the boundary point over the distance from the wing surface to the first row of vortices on the tube.

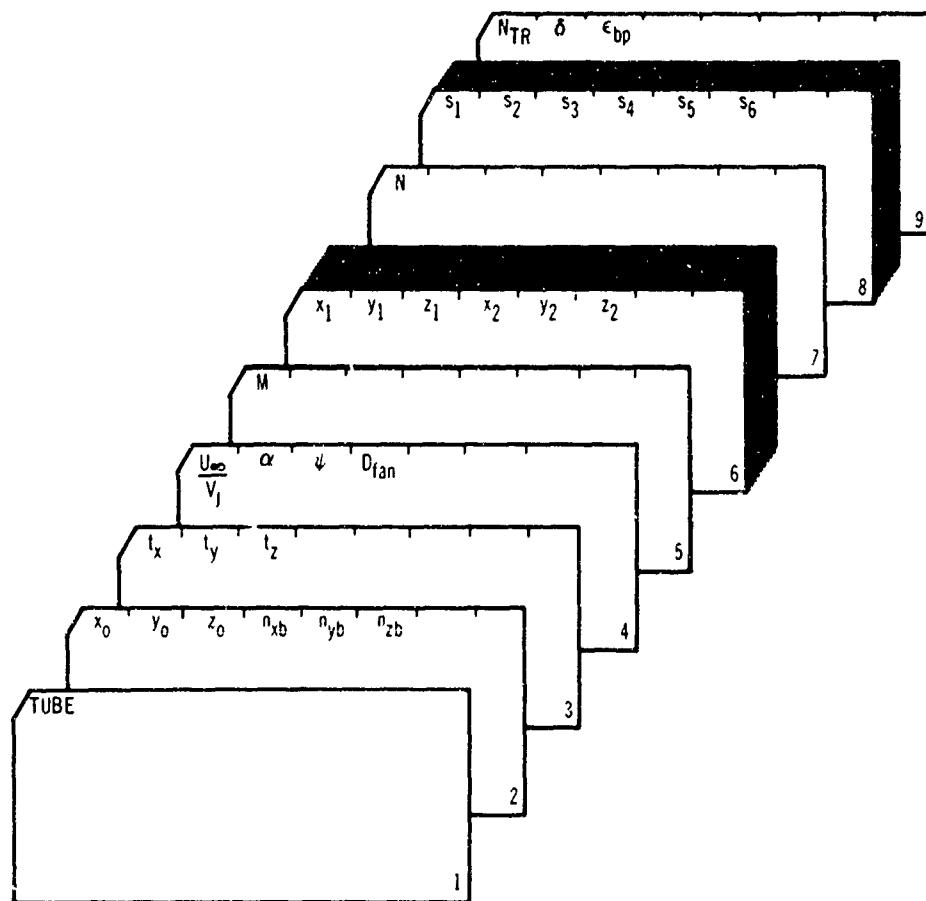


Figure 11. Data Card Arrangement for Subroutine TUBE.

AXISYM input. — Figure 12 shows the data card arrangement for subroutine AXISYM. The description of the card input to subroutine AXISYM follows.

	<u>Column</u>	<u>Code</u>	<u>Explanation</u>
<u>Card 1</u>	1-6	AXISYM	Control card—contains the word AXISYM.
<u>Card 2</u>	1-10	M	= number of pairs of (R, Z) coordinates that follow on card set 3  1. $\leq M \leq 50$ .
	11-20	$N_1$	= 1.; one (R, Z) card set follows and is used for all $\theta$ stations input.  = $N_2$ ; a separate (R, Z) card set is input for each $\theta$ station, for a total of $N_2$ (R, Z) card sets.  Note: 1. and $N_2$ are the only permissible values of $N_1$ .
<u>Card Set 3</u>	1-10	$R_1$	(R, Z) coordinates defining the contour of an axisymmetric or pseudoaxisymmetric body in a radial cutting plane. There must be M of these pairs, six numbers per card. If $N_1 = 1.0$ , one (R, Z) card set is input. If $N_1 = N_2$ , $N_2$ (R, Z) card sets must be input. This is the only permissible number of (R, Z) card sets.
	11-20	$Z_1$	
	21-30	$R_1$	
	31-40	$Z_2$	
	41-50	$R_2$	
	51-60	$Z_3$	
		$R_3$	
		$Z_3$	
<u>Card 4</u>	1-10	$N_2$	= number of radial planes that follow in card set 5  1. $\leq N_2 \leq 50$ .
<u>Card Set 5</u>	1-10	$\theta_1$	$\theta$ values of the radial planes that are used to find panel corner points in these planes. There must be $N_2$ of these values, six numbers per card.
	11-20	$\theta_2$	
	21-30	$\theta_3$	
	31-40	$\theta_4$	
	41-50	$\theta_5$	
	51-60	$\theta_6$	
<u>Card 6</u>	1-10	$x_o$	coordinates of the origin of the $(x_2, y_2, z_2)$ coordinate system with respect to $(x_3, y_3, z_3)$ reference coordinate system
	11-20	$y_o$	
	21-30	$z_o$	
<u>Card 7</u>	1-10	$a_{11}$	direction cosines of $x_2$ with respect to the $(x_1, y_1, z_1)$ coordinate system
	11-20	$a_{12}$	
	21-30	$a_{13}$	

	<u>Column</u>	<u>Code</u>	<u>Explanation</u>
<u>Card 8</u>	1-10 11-20 21-30	$a_{21}$ $a_{22}$ $a_{23}$	the direction cosines of $y_2$ with respect to the $(x_1, y_1, z_1)$ coordinate system
<u>Card 9</u>	1-10 11-20 21-30	$a_{31}$ $a_{32}$ $a_{33}$	the direction cosines of $z_2$ with respect to the $(x_1, y_1, z_1)$ coordinate system
<u>Card 10</u>	1-10 11-20 21-30	$k_1$ $k_2$ $k_3$	the distortion factors that multiply respectively the $(x_3, y_3, z_3)$ panel corner-point coordinates. The coordinates are distorted after they have been transformed into the $(x_3, y_3, z_3)$ coordinate system.
<u>Card 11</u>	1-10 11-20 21-30	$x_p$ $y_p$ $z_p$	the displacement factors that are added to the $(x_4, y_4, z_4)$ panel corner-point coordinates that have been previously distorted by $(k_1, k_2, k_3)$
<u>Card 12</u>	1-10	OUTCODE	= 0. ; the network of panel corner points will be output with the $\theta$ varying first. = 1. ; the $(R, Z)$ will vary first in the output.

Note: It is important to determine beforehand which OUTCODE is desired, since this determines whether the coordinate output defines a left-hand or a right-hand network.

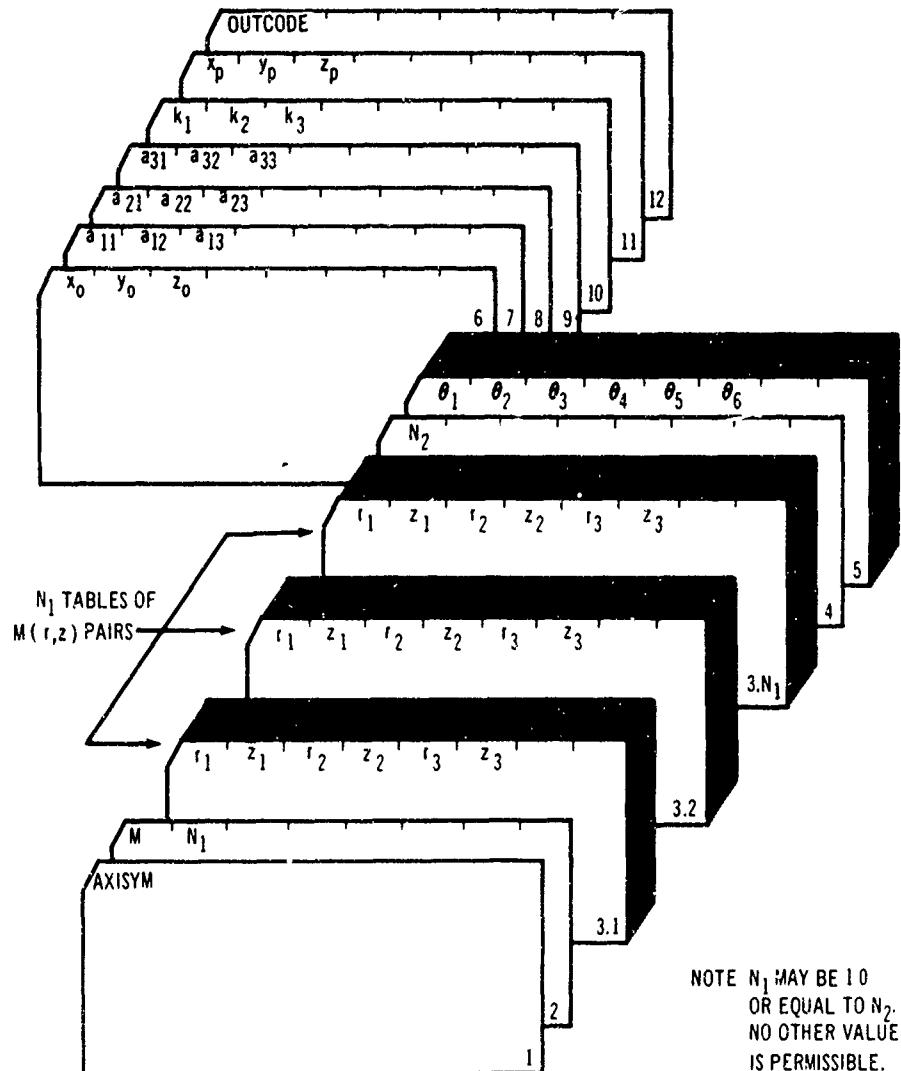


Figure 12. Data Card Arrangement for Subroutine AXISYM.

When variable spacing is used for the quadrilateral vortices and boundary-point locations on the tube, as described in Volume I, Section 6.4, the TUBE subroutine is used to generate the positions of the tube boundary points, as well as the vortices. This is an optional feature of the potential-flow program; ordinarily, it computes the position of the boundary point for the quadrilateral vortices on the tube, although this feature should not be used with variable tube spacing. The procedure for generating the vortex boundary points is outlined below.

First, a new  $s_k$  table for the boundary-point locations is used to input to subroutine TUBE.

Second, a new array of  $(x, y, z)_i$  initial points for input to subroutine TUBE is prepared by averaging adjacent x, y, and z values of the original initial points on the wing lower surface. Since for an original tube there are  $M \times N$  panel corner points, there are  $(M - 1) \times (N - 1)$  boundary points. For the tube boundary-point case, there is one less arc length and initial point than for the original tube geometry.

The third step is the card output procedure. A nonstandard print format is used by punching (3F10.5) in card columns 11-18 of the CARD control card preceding the tube data cards. The boundary points are thus punched out three per card rather than the normal six per card, and will be in the format required by the potential-flow program.

### 3.1.3 Monitor Control Cards

Figure 13 displays the monitor control cards necessary to execute the geometry program. A detailed description of each card follows:

Monitor Control Card 1 is the job card, containing the job name, priority, central processor time limit, and central memory requirement. Fields are separated by commas, the last field being terminated by a period.

NAME is the job name, one to seven characters; the first character must be a letter. The SCOPE Operating System replaces characters 5, 6, and 7 with a unique sequence number indicating the number of jobs run through the system since dead start.

PR is the job priority, 1 through 17 (octal).

T is the central processor time limit in seconds (octal).

FL is the field length for the job (octal).

Monitor Control Card 2 is the accounting card, distinguished by the letters A, C, C, and T in columns 1 through 4. Information on this card will vary between computer installations, but generally includes the user's name, location, etc.

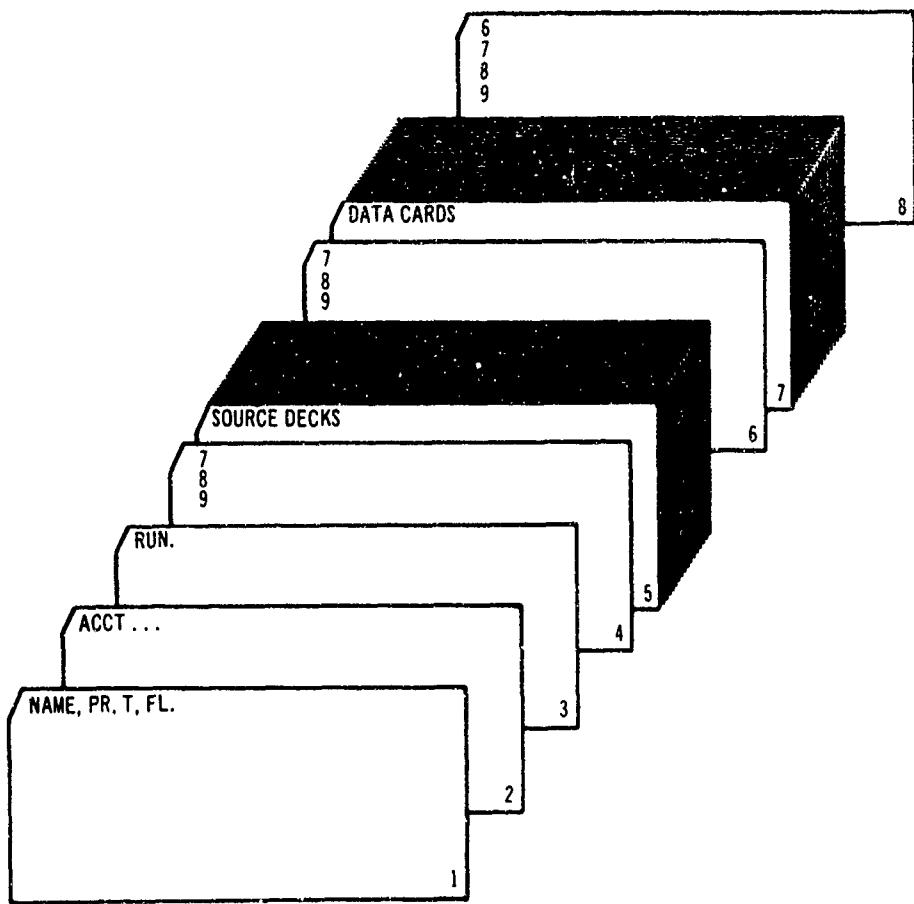


Figure 13. Monitor Control Card Arrangement for Geometry Program.

Monitor Control Card 3 is the program call card. In this case, the RUN compiler is called into core to compile and execute the source decks. The letters R, U, and N are in columns 1, 2, and 3, followed by a period in column 4.

Monitor Control Card 4 is an end-of-record card. This card contains 7-8-9 punches in column 1.

Card Set 5 represents the source decks. Note that there is a terminal end-of-record card (monitor control card 6), but that there are no end-of-record cards between individual source decks.

Monitor Control Card 6 is an end-of-record card.

Card Set 7 represents the input data. These cards will vary, depending on the particular problem to be run.

Monitor Control Card 8 is an end-of-file card. It contains 6-7-8-9 punches in column 1. There must be only one of these per deck, and it is always the last card.

### 3.1.4 Program Field Length, Timing, and Output Estimate

A field length of 104,000<sub>8</sub> is required for compilation and execution of the geometry program.

The program requires approximately 20 seconds of central processor time to process one surface type, except the jet efflux tube, which will require about 1 minute. The number of lines printed depends on the number and size of the cases and is approximately six times the number of panels.

## 3.2 POTENTIAL-FLOW PROGRAM

### 3.2.1 Machine Components

The potential-flow program is coded in the FORTRAN IV and ASCENT languages for the CDC 6600 (131k) digital computer. Control of the computer is monitored by the SCOPE Operating System. The program is organized to utilize the segmentation feature of the loader.

The program requires 20 data files. They consist of one input data file, one output file, one punch data file, and 17 scratch data files. The data files may be stored on disks or magnetic tapes.

### 3.2.2 Input Data Format

This section describes the usage of the potential-flow program. The first part provides background information and introduces the nomenclature used in the preparation of input data. The second part is a detailed description of the card input format.

Background information.

Reference coordinate system. — All geometric inputs must be in the x, y, z reference coordinate system. They can be given in any convenient length dimension. The location and the size of the configuration in this coordinate system is arbitrary except when the symmetry option is used. The angles of attack and yaw define the free-stream direction with respect to the reference coordinate axes. For  $\alpha = \psi = 0^\circ$  the free-stream velocity is directed along the positive x-axis of the reference coordinate system.

Velocities. — All velocities used in the program are nondimensionalized with respect to the free-stream velocity, except for the special case of zero free-stream velocity. In particular, specified normal velocities appearing in the boundary conditions must be nondimensional with respect to the free-stream velocity. For the special case of zero free-stream velocity, the boundary conditions must include nonzero specified velocities, or the solution will be trivial. For this case all the velocities should be considered as velocities nondimensionalized with respect to a reference velocity  $U_R$  used to define the pressure coefficient.

$$C_p = \frac{p - p_\infty}{\frac{1}{2} \rho U_R^2} \quad (7)$$

If necessary, the resultant velocity components can be made dimensional by multiplying them by the free-stream velocity or reference velocity for the respective cases.

Symmetric and unsymmetric flow. — The potential-flow program handles two types of flow: symmetric and unsymmetric. For symmetric flow, the plane of symmetry is the x-z plane and inputs are required only for  $y \geq 0$ . The angle of yaw must be zero. With symmetry, the singularity strengths are symmetric about the x-z plane and the program combines the influence of the reflected side ( $y \leq 0$ ) with the influence of the basic side. The obvious advantage of a symmetric flow problem is that the number of singularities is only half of those required on an unsymmetric flow problem, for which the entire body must be defined.

Simultaneous solutions. — The Aerodynamic Section has the capability of producing up to five aerodynamic solutions simultaneously for a given geometric configuration. Variations of  $\alpha$ ,  $\psi$ , specified inflow distributions, and the zero free-stream condition can be run simultaneously with only slightly greater computation time than required for a single case. This economy occurs because the influence coefficient matrix, which depends only on the geometric configuration, is only computed once.

While several solutions can thus be obtained for a single run, the user must be aware of the effect of further approximations in the theoretical model that may be introduced. For example, the placement of the multihorseshoe vortices for a wake or efflux tube are assumed for a given direction of the free-stream velocity and inlet velocity ratio. Moderate variations of the free-stream direction and

inlet velocity ratio with the wake and efflux tube positions fixed will still yield useful solutions. However, large variations of these quantities may result in a poor representation of the physical problem; for example, large deviations of the wake and efflux tube from their real position in the flow. As another example, flows with zero free-stream velocity should not be run simultaneously with flows having a free stream, for the zero free-stream cases do not have a trailing wake. However, it is advantageous to combine several zero free-stream cases having different fan inflow velocity distributions.

Off-body points.—The velocity components and pressure coefficients can be found at points off the body by specifying their location in the reference coordinate system. The off-body velocity can be calculated everywhere except in the immediate vicinity (within  $10^{-5}$  units) of a source panel or vortex segment. While the above limitations reflect the computer program's ability to compute the velocity, there exist larger regions where the velocity will not be physically correct. As explained in Volume I, Section 6, velocities calculated near source-panel edges or near concentrated vortices will be meaningless.

Source and quadrilateral vortex panels.—To describe a single source panel or a quadrilateral vortex, it is necessary to input four corner points  $(x_i, y_i, z_i)$  for  $i = 1, 2, 3, 4$  in the reference coordinate system. The order of the points is shown in Figure 14(a). The order of points is important; it controls the direction of the unit normal vector computed by the program at the panel boundary point, and also dictates on which face of a source panel the boundary condition is satisfied. The unit normal vector for this panel would be generated as  $(\bar{23} \times \bar{14})$ , where  $\bar{23}$  designates the vector from corner (2) to corner (3). It thus points upward when the corner points are ordered as shown. For source panels, the boundary condition is imposed on the face containing the outward directed normal, so it is necessary that all normals on a source-paneled surface be directed outward.

A group of panels or quadrilateral vortices can be described more conveniently as a panel network, defined by a rectangular array of panel corner points. The ordering of the point array is shown by the parenthetical numbers in Figure 14(b). These points are arranged in  $N$  columns, each containing an equal number of points. The size of the network is designated as  $M \times N$ , the total number of points. The total number of panels is  $(M - 1) \times (N - 1)$ . In the example of Figure 14(b),  $M = 4$ ,  $N = 5$ , the size of the network is  $M \times N = 20$ , and the number of panels is  $(M - 1) \times (N - 1) = 12$ .

The terminology used to describe the position of a corner point or singularity panel in a network is shown in Figure 15. Each is denoted in terms of its position in a particular column. A singularity panel in Figure 15 bears the same column designation and position in the column as its lower left corner point. There is one less singularity panel column than point column and one less singularity per column than point per column. A network arranged on a surface as shown is called a right-hand network because when viewed by an observer standing on the surface exterior and facing in the direction of increasing position in a column, the column number increases to the right. It is necessary that all source-panel networks be right-handed, so that the surface normals constructed by the program will be directed outward. For consistency it is helpful, but not necessary, to input quadrilateral vortex networks in the same

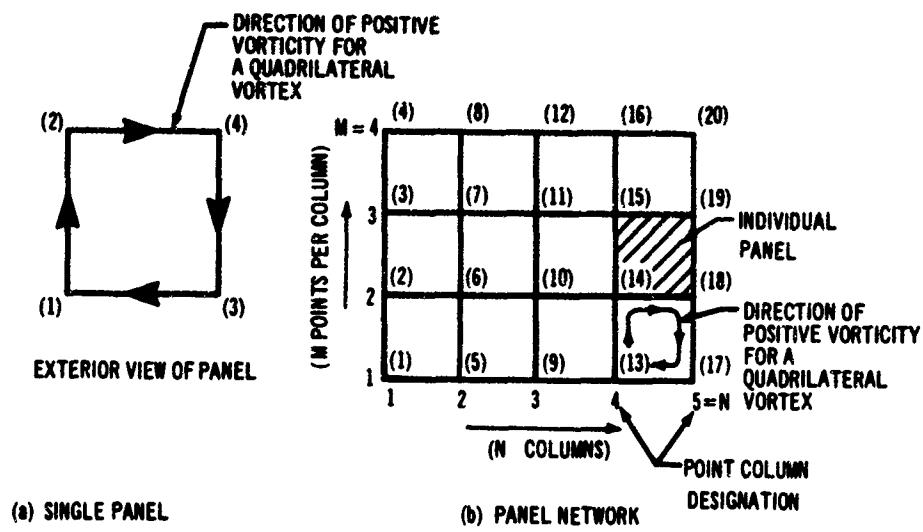


Figure 14. Source and Quadrilateral Vortex Networks.

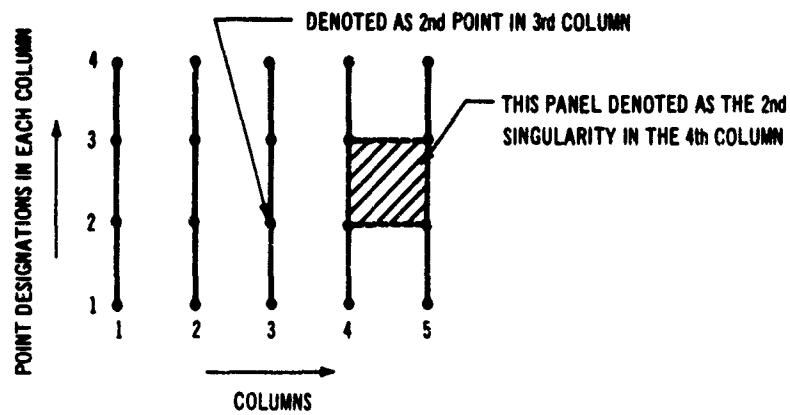


Figure 15. Column Designations.

manner so that all normals point outward from the surface. The direction of positive vorticity for a single quadrilateral vortex is shown in Figure 14(a). Similarly, each quadrilateral vortex in the network of Figure 14(b) is positive when directed clockwise.

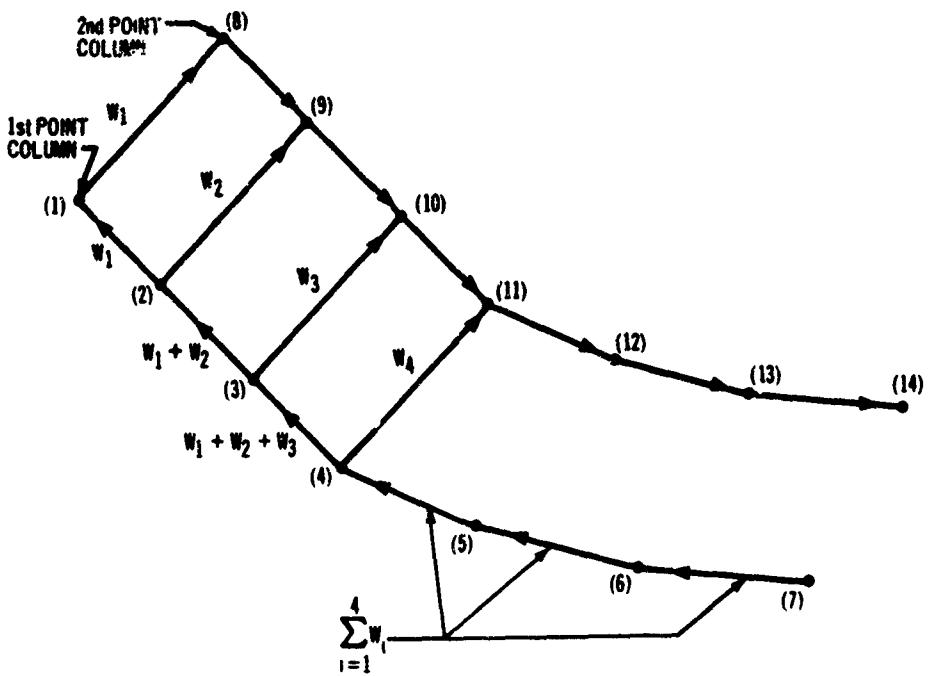
Multihorseshoe vortex. — The properties of a multihorseshoe vortex are defined in Volume I, Sections 3 and 6. A single multihorseshoe vortex singularity is input by specifying two columns of points in the reference coordinate system, the number of points in a column (both columns must have the same number of points), the number of bound vortex segments MS, and the weighting  $W_i$  ( $i = 1, \dots, MS$ ) of the bound vortex segments. The order of the points for inputting is shown in parenthetical numbers in Figure 16(a). The bound vortex segments are formed by connecting the first MS points in both columns. The direction of positive vorticity for positive weighting is from point column one to point column two.

For a multihorseshoe vortex with trailing legs, such as shown in Figure 16(a), there is no restriction on the choice of weights. The vortex strength of each bound element will be equal to the singularity strength  $\sigma$ , multiplied by the weight of the bound element. The vorticity of the trailing elements follows from the continuity of vorticity requirement. It is often convenient to select the weights such that  $\sum_i W_i = 1$ , so that the total circulation of the multihorseshoe will be equal to  $\sigma$ . It is essential that the points at the ends of the trailing segment (points 7 and 14 in this example) be remote from the wing, since the trailing legs represent vorticity extending to infinity.

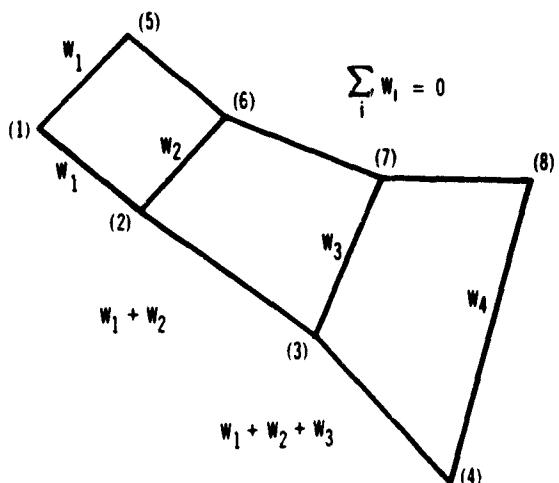
The form of multihorseshoe vortex used for the fan and tube internal systems is shown in Figure 16(b). The number of bound elements is equal to the number of points in a column, and there are no trailing legs. To maintain continuity of vorticity, it is necessary that  $\sum_i W_i = 0$ . If this requirement is not met, the

resulting solution will not satisfy Laplace's equation. An example of the weighting assigned to the multihorseshoe vortex in Figure 16(b) might be  $W_1 = -0.4$ ,  $W_2 = -0.3$ ,  $W_3 = -0.3$ ,  $W_4 = +1$ . The vorticity on the fourth bound element would then be directed opposite to the other three, with strength equal to the sum of the other bound vortices. As explained later, the weights applied to the fan and tube internal systems usually are arranged in a similar manner. The one bound element that is exterior to the wing (forming either the first circumferential vortex in the fan face or the first segment of the efflux tube) is assigned a weight of +1, and the sum of the other bound elements interior to the wing must then be -1.

A group of multihorseshoe vortices with common sides are conveniently described by a multihorseshoe network, as shown in Figure 17. The parenthetical numbers display the ordering of the input coordinates. The column concept of points is adapted here also, with N denoting the number of columns and M the number of points in each column. A network is described by the numbers M and N, the total number of points  $M \times N$ , the number of bound segments (MS) in a column, and the weights  $W_i$  ( $i = 1, \dots, MS$ ). One set of weights may be applied throughout a network, or the weighting of each multihorseshoe may be assigned separately. The former option is generally used. For the example in Figure 17,  $M = 5$ ,  $N = 6$ ,  $MS = 3$ . The total number of multihorseshoes is  $N - 1 = 5$ .



(a) SINGLE MULTIHORSESHOE WITH TRAILING ELEMENTS



(b) SINGLE MULTIHORSESHOE WITHOUT TRAILING ELEMENTS

Figure 16. Single Multihorseshoe Vortex Arrangements.

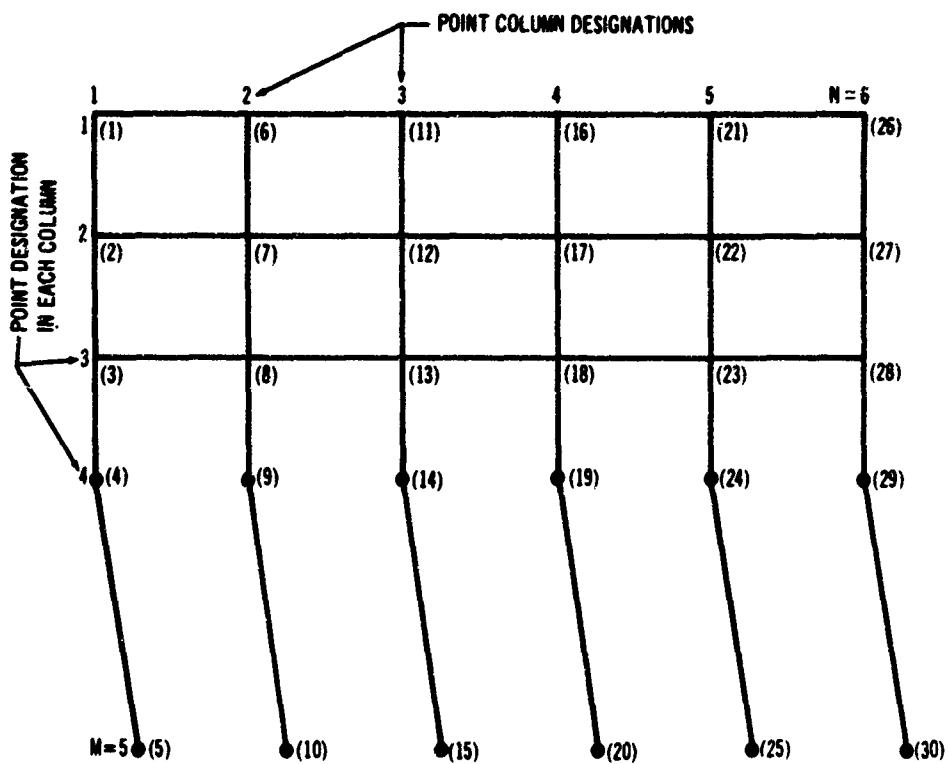


Figure 17. Multihorseshoe Vortex Network Designations.

Boundary-point conditions. — The location  $(x, y, z)$  and the unit normal vector  $(n_x, n_y, n_z)$  of the boundary points are determined differently for each of the different singularities. For a source panel, the boundary point is fixed by the program at the centroid of the plane panel, with the unit normal vector perpendicular to the panel. The quadrilateral vortex boundary-point location and normal can be computed by the program or input to the program. The computed location of the boundary point is the average of the input points that define the quadrilateral vortex, and the unit normal vector is directed along the vector product of the diagonals of the quadrilateral vortex. Boundary-point conditions for the multihorseshoe vortex must be input to the program. The various input options for a network are shown in Table I.

The program recognizes two types of boundary conditions at the boundary points. The first boundary condition requires zero normal flow at a boundary point; that is, the flow must be tangential to the surface. The second condition requires a specified velocity component  $U_s$  in the direction of the unit normal vector. If unit normal vectors point outward from a surface,  $U_s$  will be negative for flow into a surface.

**TABLE I. BOUNDARY-POINT CONDITIONS**

Singularity	Computed by Program	Input to Program
Source Panel	x y z $n_x \ n_y \ n_z$	-
Quadrilateral Vortex	x y z $n_x \ n_y \ n_z$	-
	-	x y z $n_x \ n_y \ n_z$
	$n_x \ n_y \ n_z$	x y z
	x y z	$n_x \ n_y \ n_z$
Multihorseshoe Vortex	-	x y z $n_x \ n_y \ n_z$

**Forces and moments.**—The forces and moment coefficients on both the external wing surface and the lift fans can be calculated as an option in the program. The necessary reference quantities to be input are the coordinates of a point ( $x_r, y_r, z_r$ ) at the origin of the moment axes, the reference planform area  $S_r$ , chord  $c_r$ , and span  $b_r$ .

**Lift-fan barrier.**—The assembly of singularity networks (multihorseshoe and quadrilateral vortex) representing the fan face and associated internal systems inside the wing and centerbody is termed the barrier assembly. In the framework of the program, each barrier assembly can be represented by up to seven singularity networks. Each of these singularity networks must be identified in the input as belonging to a barrier assembly.

All networks in the barrier assembly must be input in a specific manner if correct fan forces and velocities on the fan face are to be obtained. The point columns of each network must be radially oriented, with the column designation increasing counterclockwise when viewed from above. Each singularity network must evolve completely around the fan axis, 360 degrees, except for a fan with its axis in the plane of symmetry. For the latter case, each network evolves half way around the fan axis. The point columns or radial elements of all singularity networks must line up with one another, and all must begin and end at the same angular position, which is arbitrary. All network corner points not interior to the wing or centerbody must lie in the plane of the fan face. In addition, each network must be input in a manner such that the direction of positive vorticity for each individual singularity (quadrilateral or multihorseshoe) is clockwise.

when viewed from above. Finally, the individual networks must be ordered sequentially in the order of decreasing distance from the fan axis.

In practice, a barrier is usually represented by a combination of three networks: two quadrilateral and one multihorseshoe. The location of a column of each network with respect to the inlet and centerbody is shown in Figure 18(a). The multihorseshoe network forms the internal lifting system associated with the barrier (see Volume I, Section 6.4) and extends to the outermost circumferential segment on the barrier. The first quadrilateral network is located entirely on the barrier. The second quadrilateral network extends from the innermost circumferential segment on the barrier to the center of the centerbody. Each network extends entirely around the barrier, 360 degrees. The three networks may be input in any order, but must be assigned the following sequence: (1) the multihorseshoe, (2) quadrilateral 1, and (3) quadrilateral 2.

The rules for input arrangement laid down above are satisfied when these three networks are input in the following manner. The multihorseshoe network may be input in either of the two ways specified in Figure 18(b) and (c). In either case, the network point columns are radially oriented, with the column designation increasing counterclockwise as viewed from above. In both cases, the weight of the segment on the barrier must be +1. A weight of zero is applied to the point of intersection of the radial vortices with the inlet wall to allow a bend in the vortices at this point. The remainder of the weights of the segments internal to the wing must add up to -1. The number of internal bound elements used is optional. The boundary points for this system are placed adjacent to the source panels on the inlet wall (see Volume I, Section 6.4).

The first quadrilateral network on the barrier is input in the order shown in Figure 18(d). The vortex spacing must follow the recommendations given in Volume I, Section 6.4. If uniform spacing is used radially, the automatic boundary-point placement feature of the quadrilaterals can be used for the boundary-point placement.

The second quadrilateral network is input as shown in Figure 18(e). The point columns are radial, as before, with the column designation increasing counterclockwise. The boundary-point coordinates for the second quadrilateral must be input.

A systematic specification of the direction of the unit normal vectors at the barrier boundary points should be followed so that the normal velocity is properly directed. If all vectors are directed up, then the specified normal velocity  $U_s$  at the barrier boundary points must be negative for flow into the fan. If all vectors are directed down into the fan, the specified normal flow must be positive. Use of the automatic boundary-point placement feature with the first quadrilateral network will produce upward-pointing unit vectors; their direction may be reversed, if desired, by the program option. Some of the above requirements are summarized in Table II.

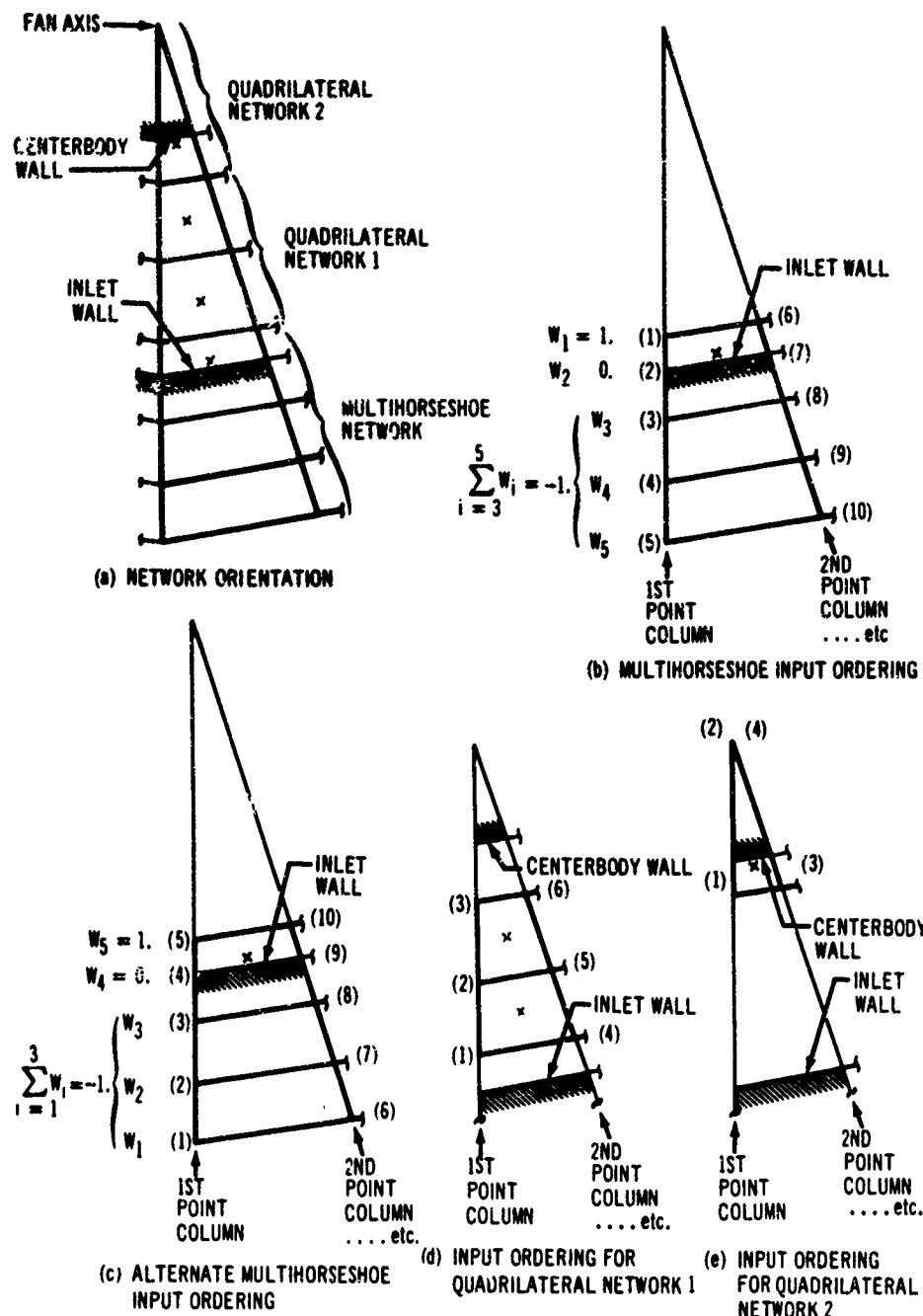


Figure 18. Barrier Networks.

TABLE II. SUMMARY OF BOUNDARY-POINT PLACEMENT REQUIREMENTS

Vortex Networks	Network Size	Boundary-Point Location	Unit Normal Up*	Unit Normal Down**
Multihorseshoe	$M_1 \times N$	Input	Input	Input
Quadrilateral 1	$M_2 \times N$	Computed (if uniform radial spacing is used)	Computed	Input
Quadrilateral 2	$2 \times N$	Input	Input	Input

\* $U_g$  will be negative for flow into the fan.  
 \*\* $U_g$  will be positive for flow into the fan.

Additional input information must be furnished for the computation of velocities on the barrier, barrier areas, and fan forces. These input data consist of the direction cosines ( $n_{xb}, n_{yb}, n_{zb}$ ) of the fan axis, directed upward, the coordinates ( $x_b, y_b, z_b$ ) of the point of intersection of the fan axis with the barrier plane, and a table of radii ( $r_i$ ) and angles ( $\theta_j$ ). The average fan exit pressure coefficient  $C_{pe}$ , the centerbody base pressure  $C_{pc}$ , the diameter  $d_c$  of the centerbody base, the distance  $h$  between the barrier and the fan exit plane, and the direction cosines ( $t_x, t_y, t_z$ ) of the vectored fan exit flow must also be specified. The radii table is composed of the radial distances from the fan axis to the inlet wall, the singularity corner points, and the centerbody wall. The table of angles  $\theta_j$  denotes the angular position of the radial barrier vortices.  $\theta$  is referenced from a radial line in the barrier, parallel to the  $x$ - $z$  plane and directed aft from the fan axis. The value of  $\theta$  is positive in the counterclockwise direction as viewed from above. The program can accept a maximum of ten lift fans for any one configuration.

Streamlines.—Streamlines can be calculated on source panels for any or all of the five solutions that may be run simultaneously. A solution is identified by its order in the input. The starting point for the backward tracking of a streamline is identified by specifying the streamline starting panel and a fractional distance along one of the panel edges where the streamline tracing begins. The initial panel is identified by means of panel column terminology, specifying its position  $i$  in a source-panel column, the column number  $j$  and network  $k$ . The particular panel edge  $e$  is numbered 1, 2, 3, or 4 according to its position in the network, as shown in Figure 19(a). The position of the starting point along an edge is specified by giving its fractional distance along the edge, measured in the direction of clockwise travel around the panel [Figure 19(b)].

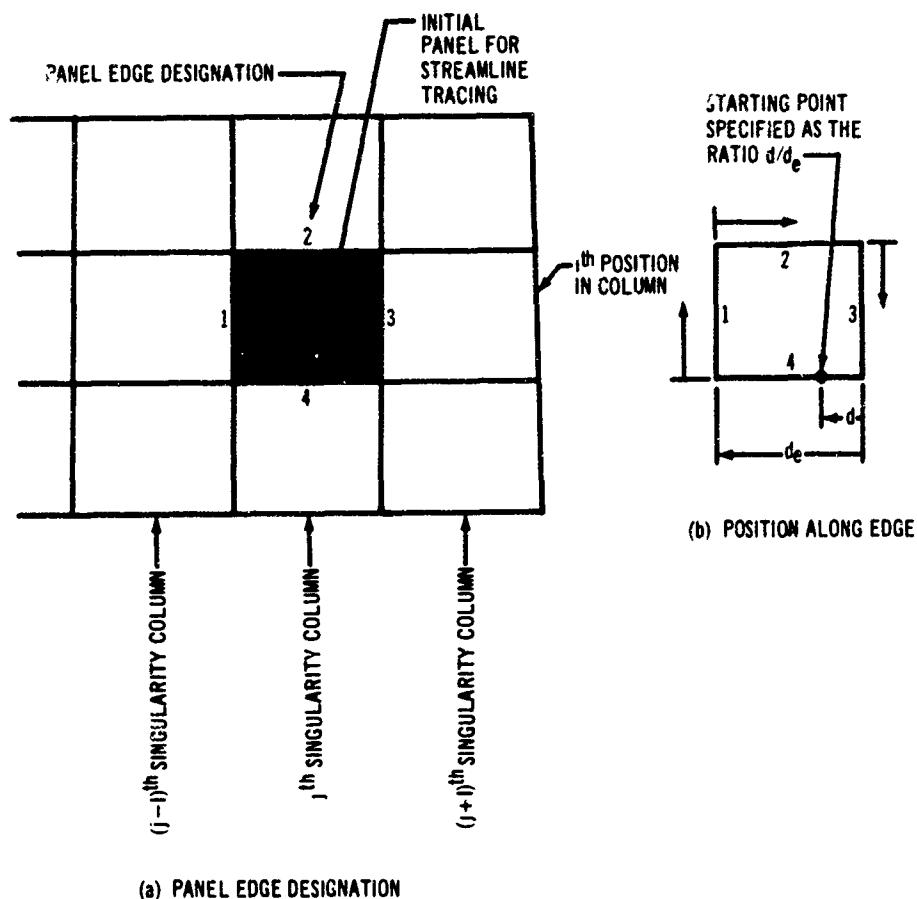


Figure 19. Initial Streamline Data.

The velocity derivative  $dV_t/ds_t$  is obtained by finite difference from a velocity calculated at a small offset distance from a source-panel boundary point. This offset distance is formed from an input fraction times the maximum source-panel diagonal of a configuration. Recommended values of the input fraction are between 0.01 and 0.0001.

Checking inputs. — If meaningless results are to be avoided, it is imperative that all geometry be input correctly. For large, complex configurations involving many networks of different singularities, it is not uncommon for errors to occur in the inputs, either through a mispunched data card, or by a simple oversight, or perhaps because of lack of program experience. The program has been designed to furnish aid in checking the inputs by means of the following feature.

The program is divided into several sections, the first one dealing with geometry. This section converts the input coordinate arrays and other information into individual panel coordinates, generates boundary-point coordinates and normals, and in general prepares all geometric information for direct use by later sections. The program can be set to run through the Geometric Section only, and then stop. It can do this extremely rapidly, the time required being approximately 1 percent of the total time required to solve the problem. The output will list the input cards, the boundary-point coordinates and normals, the panel corner points, the area and maximum diagonal of each panel, and other descriptive information listed in Volume I, Section 5.2.3. Also printed out will be the input for the Aerodynamic Section with the comment INVALID CONTROL CARD. The user can then scan these data, looking for irregularities. A mis-punched coordinate will usually result in a panel whose area or maximum diagonal is very different from its neighbor's, and it is easily recognized. The displacement distance D that a corner point is moved to produce a planar panel is also printed out, and large values of D are a definite indication that something is wrong. All boundary-point normals should be scanned to ensure that they point outward. A mistake in ordering the network corner points can easily produce this type of error.

It is strongly suggested that this check run and data scanning process be performed before committing a lengthy run to the computer. Because the time required for the check run is negligible, no data-saving feature has been included. After the output has been scanned and errors corrected, the input data must be resubmitted for running through the complete program.

#### Input format.

This section provides the instructions for assembling a data deck for the potential-flow program.

There are three types of inputs for the potential-flow program: numerical data, title card, and control cards. The numerical data are punched in seven-field, ten-digit format, and all numbers must have a decimal point. The title card will accept any alphanumeric characters. The control cards are identified from the first four columns of a card. The remainder of a control card may contain any alphanumeric symbols but generally contains the words listed below.

The control cards within the data deck are used to:

- 1) mark the beginning and the end of two blocks of data

GEOMETRIC SECTION  
END OF GEOMETRIC SECTION  
AERODYNAMIC SECTION  
END OF AERODYNAMIC SECTION

- 2) terminate the computer run after one or more cases

EXIT

- 3) identify the major divisions of data within the Geometric Section

SOURCE-PANEL GEOMETRY  
QUADRILATERAL VORTEX GEOMETRY  
MULTIHORSESHOE VORTEX GEOMETRY  
OFF-BODY POINTS

If any division of the Geometric Section is not used for a case, the corresponding control card should be omitted. The ordering of these four cards must be maintained.

For submitting a check run through the Geometric Section only, remove the END OF GEOMETRIC SECTION and the AERODYNAMIC SECTION control cards from the data deck. This run will provide a listing of all the input cards plus the output of the Geometric Section.

To run multiple cases of either the complete or the check type, stack the cases consecutively as shown in Figure 20.

The detailed card input format for the Geometry Section (source-panel geometry, quadrilateral vortex geometry, multihorseshoe vortex geometry, off-body points) and Aerodynamic Section follow.

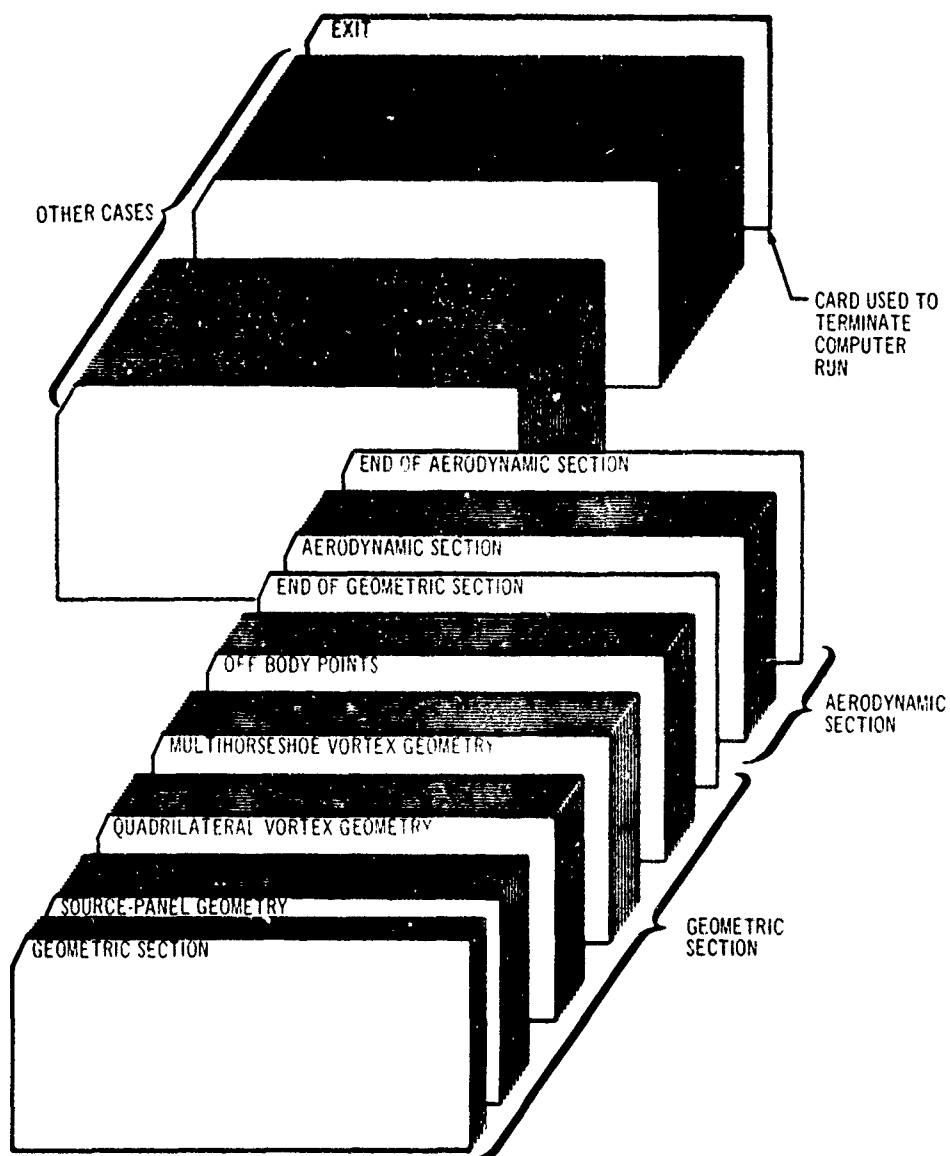


Figure 20. Data Card Arrangement for Potential-Flow Program.

### GEOMETRIC SECTION CARD INPUT

Figure 21 displays the data cards for the Geometric Section of the potential-flow program. All Geometric Section data, except title and control cards, are punched in seven-field, ten-digit format. All geometry is defined in the reference coordinate system. A description of the card input to this section follows.

	<u>Column</u>	<u>Code</u>	<u>Explanation</u>
<u>Card 1</u>	1-4	GEOM	Control card—columns 1-4 contain the word GEOM.
<u>Card 2</u>	1-80	TITLE	any desired title
<u>Card 3</u>	1-10	ICODE	= 1.; symmetric flow = 3.; unsymmetric flow
	11-20	NSING	= number of singularities used to represent the potential-flow model
			2. $\leq$ NSING $\leq$ 1200.

Source-panel geometry cards  
(See page 58.)

Quadrilateral vortex geometry cards  
(See page 60.)

Multihorseshoe vortex geometry cards  
(See page 53.)

Off-body point cards  
(See page 67.)

Card 4      1-4      END      Control card—columns 1-4 contain the word ENDb, where b represents a blank.

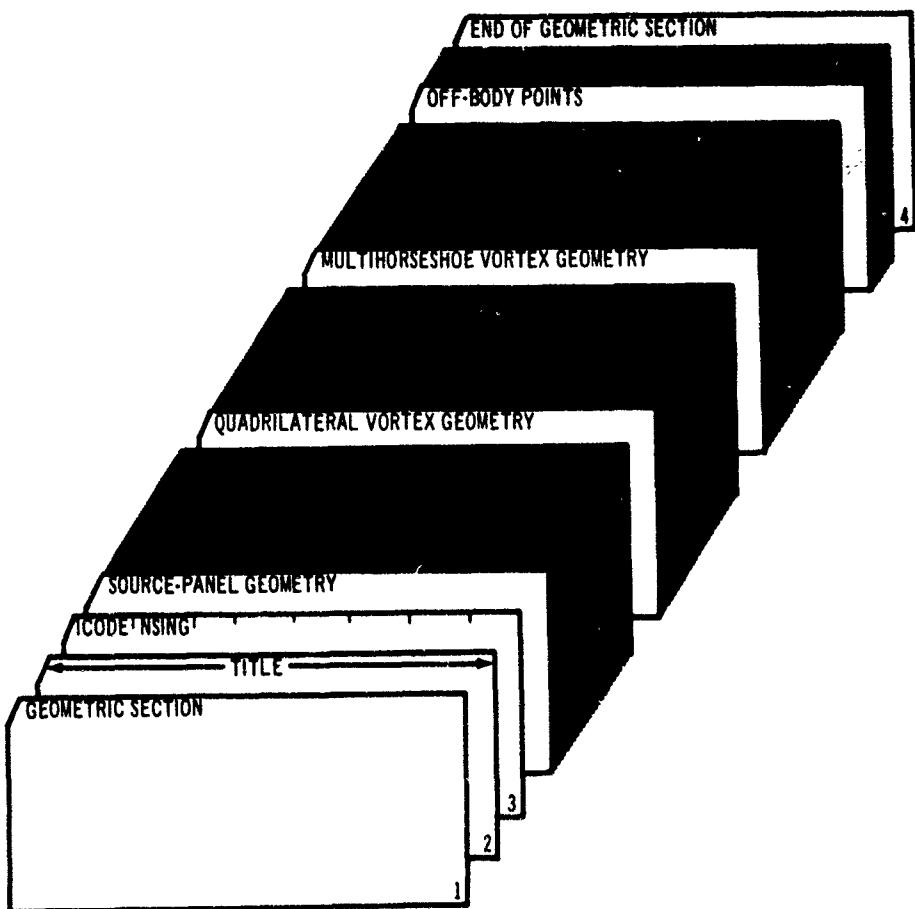


Figure 21. Data Card Arrangement for Geometric Section.

**Source-panel geometry.**—Figure 22 displays the data cards for the source-panel geometry subdivision of the Geometric Section. These cards are omitted if the potential-flow model does not require the source-panel singularity for its representation.

	<u>Column</u>	<u>Code</u>	<u>Explanation</u>
<u>Card 1</u>	1-4	SOUR	Control card—columns 1-4 contain the word SOUR.
<u>Card 2</u>	1-10	NET	= number of source-panel networks 1. $\leq$ NET $\leq$ 200.
Cards 3 through 5 are repeated for each source-panel network.			
<u>Card 3</u>	1-10	M	= number of points per column
	11-20	N	= number of point columns in the network 2. $\leq$ M $\leq$ 500., 2. $\leq$ N $\leq$ 500., and 4. $\leq$ M $\times$ N $\leq$ 2500. (See Figure 14.)
	21-30		blank
	31-40	OPTUS	= 0.; all normal velocities are zero. = 10.; normal velocities specified for each simultaneous solution. = 20.; normal velocities specified once and used for all simultaneous solutions.
<u>Card Set 4</u>	1-10	$x_1$	corner-point coordinates of the network source-panel in the reference coordinate system. The (M x N) corner points are input sequentially, two per card. (See Figure 14.)
	11-20	$y_1$	
	21-30	$z_1$	
	31-40	$x_2$	
	41-50	$y_2$	
	51-60	$z_2$	

The next card set is omitted if OPTUS = 0. The card set consists of  $((M - 1) \times (N - 1))$  cards, one for each source panel in the network, arranged in order of increasing position in successive columns (Figure 15).

If OPTUS = 10., the format of the card set is:

<u>Card Set 5</u>	1-10	$U_{s1}$	specified normal velocities, one for each simultaneous solution. The first field (columns 1-10) contains the normal velocity for the first solution, etc. The number of fields used equals the number of solutions.
	11-20	$U_{s2}$	
	21-30	$U_{s3}$	
	31-40	$U_{s4}$	
	41-50	$U_{s5}$	

If OPTUS = 20., the format of the card set is:

	<u>Column</u>	<u>Code</u>	<u>Explanation</u>
<u>Card Set 5</u>	1-10	$U_s$	specified normal velocity used for all simultaneous solutions

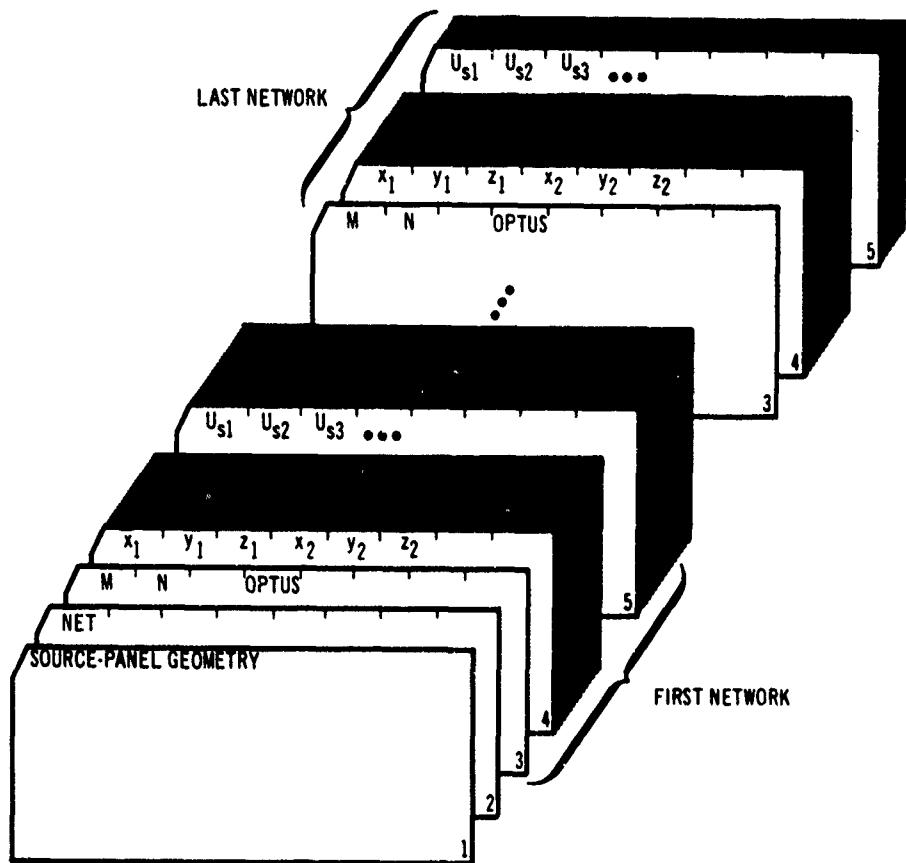


Figure 22. Data Card Arrangement for Source-Panel Geometry.

Quadrilateral vortex geometry.—Figure 23 displays the data cards for the quadrilateral vortex geometry subdivision of the Geometric Section. These cards are omitted if the potential-flow model does not require the quadrilateral vortex singularity for its representation.

	<u>Column</u>	<u>Code</u>	<u>Explanation</u>
<u>Card 1</u>	1-4	QUAD	Control card—columns 1-4 contain the word QUAD.
<u>Card 2</u>	1-10	NET	= number of quadrilateral vortex networks 1. $\leq$ NET $\leq$ 200.
Cards 3 through 6 are repeated for each quadrilateral vortex network.			
<u>Card 3</u>	1-10	M	= number of points per column
	11-20	N	= number of point columns in the network 2. $\leq$ M $\leq$ 500., 2. $\leq$ N $\leq$ 500., and 4. $\leq$ M $\times$ N $\leq$ 2500. (See Figure 14.)
	21-30	OPTBP	= 0.; boundary-point coordinates are computed by the program. = 10.; boundary-point coordinates (x, y, z) and unit normal vector components ( $n_x, n_y, n_z$ ) are to be input for all quadrilateral vortices of the network. = 20.; only the boundary-point coordinates (x, y, z) are to be input for all quadrilateral vortices of the network (unit normals constructed by the program). = 30.; only the boundary-point unit normal vector components ( $n_x, n_y, n_z$ ) are to be input for all quadrilateral vortices of the network (boundary-point coordinates computed by the program).
	31-40	OPTUS	= 0.; all normal velocities are zero. = 10.; normal velocities specified for each simultaneous solution. = 20.; normal velocities specified once and used for all simultaneous solutions.

	<u>Column</u>	<u>Code</u>	<u>Explanation</u>
<u>Card Set 4</u>	1-10	$x_1$	
	11-20	$y_1$	
	21-30	$z_1$	
	31-40	$x_2$	
	41-50	$y_2$	
	51-60	$z_2$	

The next card set is omitted if OPTBP = 0. The card set consists of  $[(M - 1) \times (N - 1)]$  cards, one for each quadrilateral vortex in the network, arranged in the order of increasing position in successive columns (Figure 15).

If OPTBP = 10., the format of the card set is:

<u>Card Set 5</u>	1-10	x	coordinates of the boundary point
	11-20	y	
	21-30	z	
	31-40	$n_x$	
	41-50	$n_y$	unit normal vector components at the
	51-60	$n_z$	boundary point

If OPTBP = 20., the format of the card set is:

<u>Card Set 5</u>	1-10	x	coordinates of the boundary point
	11-20	y	
	21-30	z	

If OPTBP = 30., the format of the card set is:

<u>Card Set 5</u>	1-30	blank	
	31-40	$n_x$	unit normal vector components at the
	41-50	$n_y$	boundary point
	51-60	$n_z$	

The next card set is omitted if OPTUS = 0. The card set consists of  $[(M - 1) \times (N - 1)]$  cards, one for each quadrilateral vortex in the network, input arranged in the order of increasing position in successive columns (Figure 15).

If OPTUS = 10., the format of the card set is:

<u>Card Set 6</u>	1-10	$U_{s1}$	array of specified normal velocities, one
	11-20	$U_{s2}$	for each simultaneous solution. The first
	21-30	$U_{s3}$	field (columns 1-10) contains the normal
	31-40	$U_{s4}$	velocity for the first solution, etc. The
	41-50	$U_{s5}$	number of fields used equals the number

If OPTUS = 20., the format of the card set is:

	<u>Column</u>	<u>Code</u>	<u>Explanation</u>
<u>Card Set 8</u>	1-10	$U_s$	specified normal velocity used for all simultaneous solutions

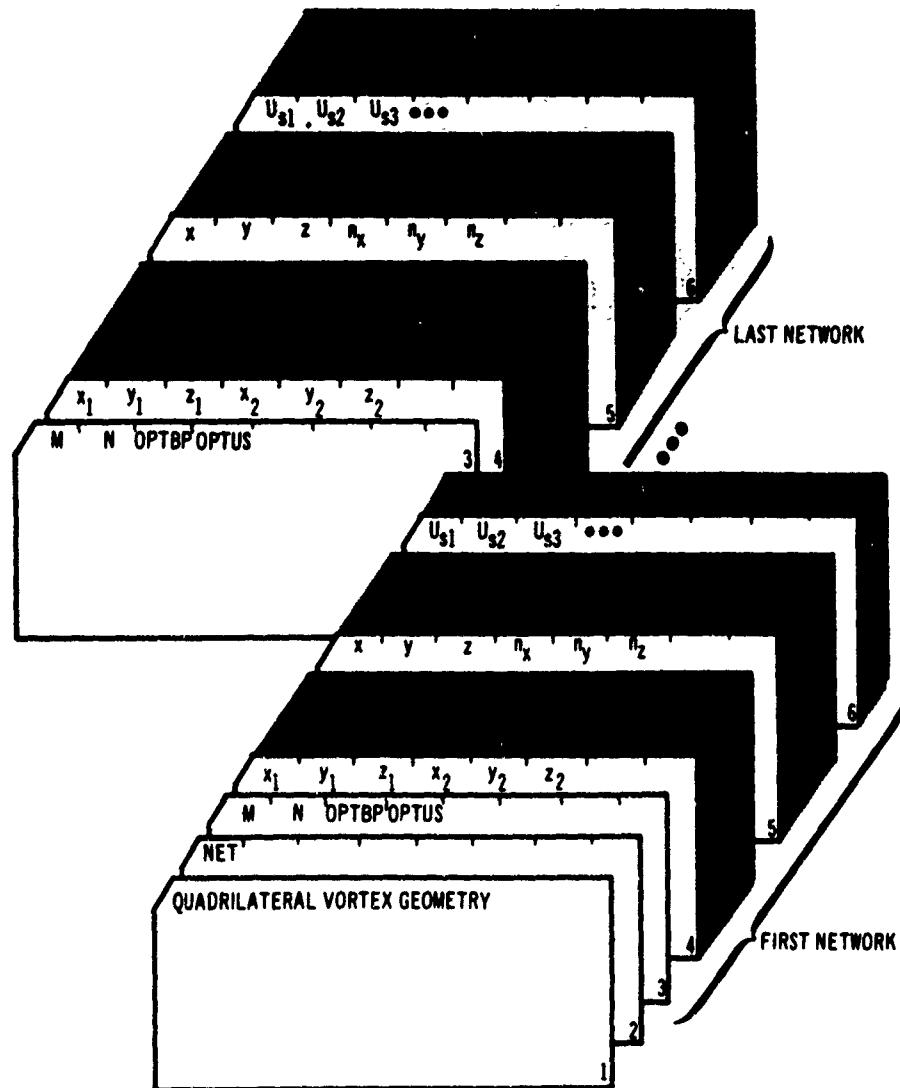


Figure 23. Data Card Arrangement for Quadrilateral Vortex Geometry.

Multihorseshoe vortex geometry. —Figure 24 displays the data cards for the multihorseshoe vortex geometry subdivision of the Geometric Section. These cards are omitted if the potential-flow model does not require the multihorseshoe vortex singularity for its representation. A maximum of 110 multihorseshoe vortex singularities may be used.

	<u>Column</u>	<u>Code</u>	<u>Explanation</u>
<u>Card 1</u>	1-4	MULT	Control card—columns 1-4 contain the word MULT.
<u>Card 2</u>	1-10	NET	= number of multihorseshoe vortex networks 1. $\leq$ NET $\leq$ 100.
Cards 3 through 7 are repeated for each multihorseshoe vortex network.			
<u>Card 3</u>	1-10	M	= number of points per column
	11-20	N	= number of point columns in the network 2. $\leq$ M $\leq$ 50., 2. $\leq$ N $\leq$ 100., and 4. $\leq$ M $\times$ N $\leq$ 2500.
	21-30	MS	= number weighted segments per multihorseshoe 1. $\leq$ MS $\leq$ 50. (See Figure 16.)
	31-40	OPTWT	= 1.; array of weighting values are specified once, and these values apply to all multihorseshoe vortices of the network. = 10.; array of weighting values are specified for each multihorseshoe vortex of the network.
	41-50	OPTBP	= 10.; boundary-point coordinates (x, y, z) and unit normal vector components ( $n_x$ , $n_y$ , $n_z$ ) are to be input for all multihorseshoe vortices of the network.
	51-60	OPTUS	= 0.; all normal velocities are zero. = 10.; normal velocities specified for each simultaneous solution. = 20.; normal velocities specified once and used for all simultaneous solutions.

	<u>Column</u>	<u>Code</u>	<u>Explanation</u>
<u>Card Set 4</u>	1-10	$x_1$	
	11-20	$y_1$	
	21-30	$z_1$	
	31-40	$x_2$	
	41-50	$y_2$	
	51-60	$z_2$	

corner-point coordinates of the multi-horseshoe vortex network in the reference coordinate system. ( $M \times N$ ) corner points are input sequentially, two per card.  
(See Figure 17.)

The next card set is for the specified weights.

If OPTWT = 1., the format of the card set is:

<u>Card Set 5</u>	1-10	$w_1$	weighting values for the bound multi-horseshoe vortex segments. MS values are input. These values apply to all multihorseshoe vortices of the network.
	11-20	$w_2$	
	21-30	$w_3$	
	31-40	$w_4$	
	41-50	$w_5$	
	51-60	$w_6$	
	61-70	$w_7$	

If OPTWT = 10., the format of the card set is:

<u>Card Set 5</u>	1-10	$w_1$	weighting values for the bound horseshoe vortex segments. ( $N - 1$ ) card sets of MS weighting values are input, one card set for each multihorseshoe vortex.
	11-20	$w_2$	
	21-30	$w_3$	
	31-40	$w_4$	
	41-50	$w_5$	
	51-60	$w_6$	
	61-70	$w_7$	

The next card set consists of ( $N - 1$ ) cards, one for each multihorseshoe vortex in the network.

<u>Card Set 6</u>	1-10	$x$	coordinates of the boundary point
	11-20	$y$	
	21-30	$z$	
	31-40	$n_x$	unit normal vector components at the boundary point
	41-50	$n_y$	
	51-60	$n_z$	

The next card set is omitted if OPTUS=0. The card set consists of ( $N - 1$ ) cards, one for each multihorseshoe vortex in the network.

<u>Column</u>	<u>Code</u>	<u>Explanation</u>
---------------	-------------	--------------------

If OPTUS = 10., the format of the card set is:

<u>Card Set 7</u>	1-10 $U_{s1}$	array of specified normal velocities, one for each simultaneous solution. The first field (columns 1-10) contains the normal velocity for the first solution, etc. The number of fields used equals the number of solutions (maximum of five).
	11-20 $U_{s2}$	
	21-30 $U_{s3}$	
	31-40 $U_{s4}$	
	41-50 $U_{s5}$	

If OPTUS = 20., the format of the card set is:

<u>Card Set 7</u>	1-10 $U_s$	specified normal velocity used for all simultaneous solutions
-------------------	------------	--

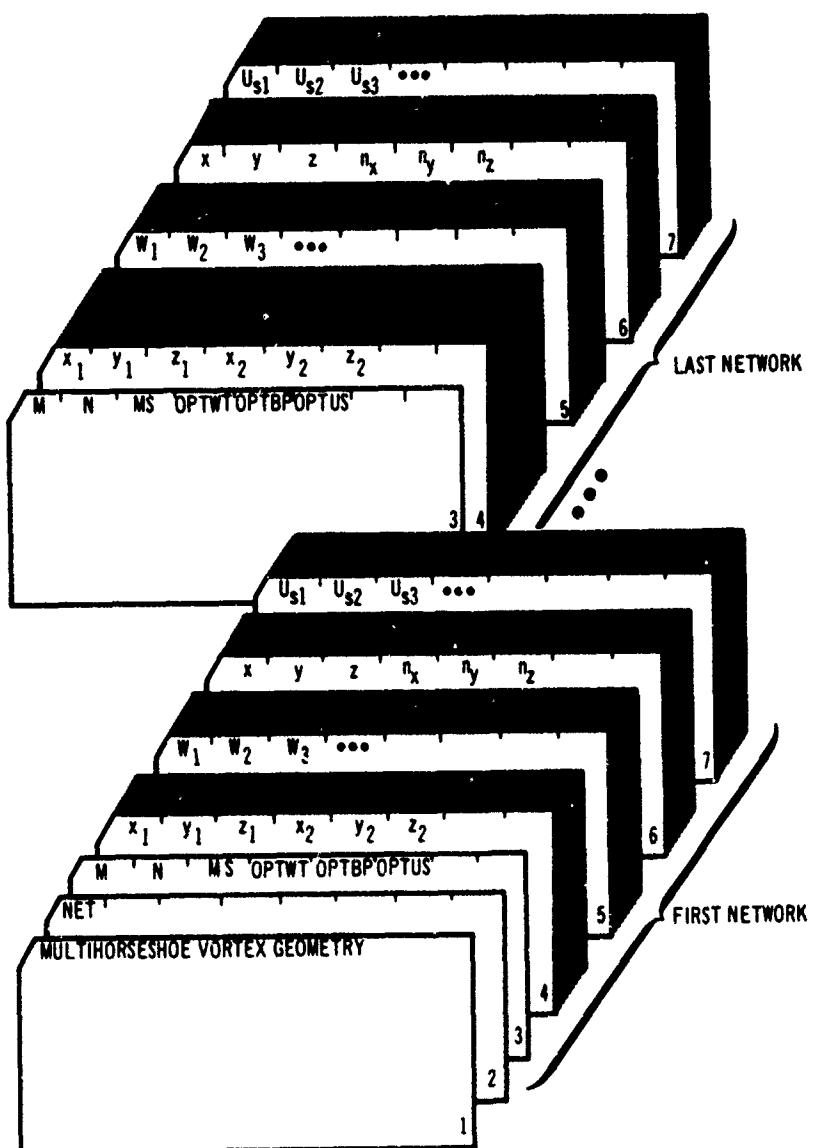


Figure 24. Data Card Arrangement for Multihorseshoe Vortex Geometry.

Off-body point geometry. —Figure 25 displays the data cards for the off-body points. These cards are omitted if the velocity at off-body points is not desired.

	<u>Column</u>	<u>Code</u>	<u>Explanation</u>
<u>Card 1</u>	1-4	OFF	Control card—column 1-4 contains the word OFFb, where b represents a blank or the word OFF-.
<u>Card 2</u>	1-10	NOFF	= number of off-body points 1. $\leq$ NOFF $\leq$ 2500.
<u>Card Set 3</u>	1-10 11-20 21-30 31-40 41-50 51-60	$x_1$ $y_1$ $z_1$ $x_2$ $y_2$ $z_2$	coordinates of the off-body points. NOFF points are input, two per card.

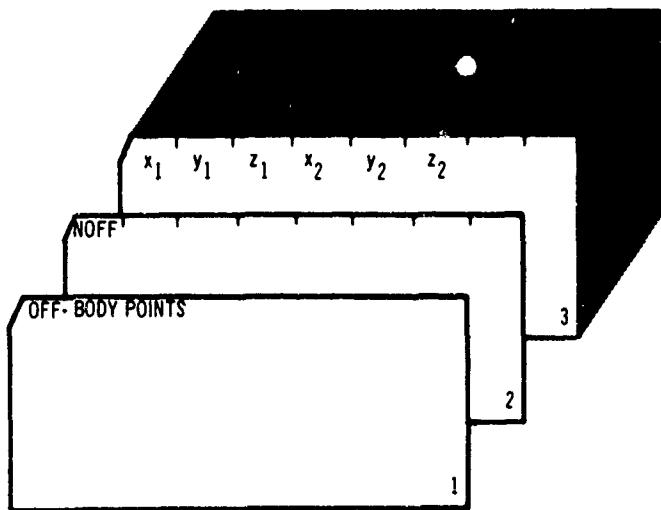


Figure 25. Data Card Arrangement for Off-Body Points.

**AERODYNAMIC SECTION CARD INPUT**

Figure 26 displays the data cards for the Aerodynamic Section of the potential-flow program.

All Aerodynamic Section data, except control cards, are punched in seven-field, ten-digit format. All input aerodynamic geometry is defined in the reference coordinate system. A description of the card input to this section follows.

	<u>Column</u>	<u>Code</u>	<u>Explanation</u>
<u>Card 1</u>	1-4	AERO	Control card—columns 1-4 contain the word AERO.
<u>Card 2</u>	1-10	NRHS	= number of simultaneous solutions desired  1. $\leq$ NRHS $\leq$ 5.
	11-20	FORCES	force and moment option code  = 0.; no forces and moments are calculated.  = 1.; all forces and moments are calculated.
	21-30	PLOT	plotting option code  = 0.; no plotting desired.  = 1.; plotting of $C_p$ vs x/c over the source-paneled model is desired (used only if plotting facility is available).
	31-40	NARRAY	= number of lift fans in the model  0. $\leq$ NARRAY $\leq$ 10.
	41-50	NSL	= number of streamlines desired  0. $\leq$ NSL $\leq$ 50.

The following card set consists of one card for each solution.

<u>Card Set 3</u>	1-10	$\alpha$	angle of attack in degrees
	11-20	$\psi$	angle of yaw in degrees

<u>Column</u>	<u>Code</u>	<u>Explanation</u>
<b>Card Set 3 (cont'd)</b>		
21-30	VZERO	zero free-stream velocity option
		= 0.; option not exercised. The dimensionless free-stream velocity has a value of unity in a direction specified by the angles of attack and yaw
		= 1.; zero free-stream velocity condition. The angles of attack and yaw have no meaning.

The following card is omitted unless FORCES = 1.0.

<u>Card 4</u>	1-10 $x_r$	moment reference axis
	11-20 $y_r$	
	21-30 $z_r$	
	31-40 $S_r$	reference planform area
	41-50 $c_r$	reference chord
	51-60 $b_r$	reference span

Cards 5 through 10 are repeated for each lift fan in the model. A maximum of ten lift fans can be specified.

<u>Card 5</u>	1-10      NET	= number of quadrilateral and multi-horseshoe networks used to form the barrier assembly  1. $\leq$ NET $\leq$ 7.
	11-20      TYPE	= 1.; half fan input. This occurs when the center of the fan is in the plane of symmetry (x-z) and the flow is symmetrical.  = 2.; complete fan input. This occurs whenever the center of the fan is not in the plane of symmetry (x-z) or when the flow is unsymmetric.
	21-30 $C_{p_c}$	assumed pressure coefficient at the centerbody base
	31-40 $C_{p_e}$	assumed pressure coefficient at the exit plane

	<u>Column</u>	<u>Code</u>	<u>Explanation</u>
<u>Card 5 (cont'd)</u>	41-50	MR	= number of radial points used at each $\theta$ location to define the barrier areas  2. $\leq MR \leq 20$ .
	51-60	NTHETA	= number of angles used to define the barrier areas  2. $\leq NTHETA \leq 50$ .
<u>Card 6</u>	1-10	$x_b$	coordinates of the barrier center
	11-20	$y_b$	
	21-30	$z_b$	
	31-40	$n_{xb}$	unit normal vector components along the fan axis, directed upward
	41-50	$n_{yb}$	
	51-60	$n_{zb}$	
<u>Card 7</u>	1-10	$t_x$	unit vector components in the direction of the flow from the fan exit
	11-20	$t_y$	
	21-30	$t_z$	
	31-40	h	average distance between the barrier and the exit plane
	41-50	$d_c$	diameter of the centerbody base
<u>Card 8</u>	1-10	$NT_1$	sequential identity of the barrier singularity networks used to form a barrier assembly. They must be ordered from the outside to the inside of the barrier.
	11-20	$NT_2$	The identification number applies to the order in which a network is input. A positive number denotes a quadrilateral vortex network, and a negative number denotes a multihorseshoe vortex network.
	21-30	$NT_3$	For example, $NT_1 = -3$ . means that the third multihorseshoe vortex network input is used to represent the first network of the barrier assembly. NET networks are listed.
	31-40	$NT_4$	
	41-50	$NT_5$	
	51-60	$NT_6$	
	61-70	$NT_7$	
<u>Card Set 9</u>	1-10	$r_1$	radial distances used to form barrier areas, in sequential order. There are MR of these.
	11-20	$r_2$	
	21-30	$r_3$	
	31-40	$r_4$	
	41-50	$r_5$	
	51-60	$r_6$	
	61-70	$r_7$	

	<u>Column</u>	<u>Code</u>	<u>Explanation</u>
<u>Card Set 10</u>	1-10 11-20 21-30 31-40 41-50 51-60 61-70	$\theta_1$ $\theta_2$ $\theta_3$ $\theta_4$ $\theta_5$ $\theta_6$ $\theta_7$	angles used to form barrier areas, in sequential order. There are NTHETA angles.
<u>Cards 11 and 12</u>			Cards 11 and 12 are repeated for each streamline on the model.
<u>Card 11</u>	1-10	KSOL	identifies the particular simultaneous solution for which the streamline is desired, based on the order of input of the simultaneous solutions
	11-20	i	1. $1 \leq KSOL \leq 5$ . position in the panel column of the streamline-starting panel
	21-30	j	panel column of the streamline-starting panel
	31-40	k	source-panel network containing the streamline-starting panel
	41-50	l	maximum length of the streamline
	51-60	$d/d_e$	fractional distance of streamline-starting point along edge e
	61-70	e	identifies the streamline-starting edge of the panel (Figure 19)
<u>Card 12</u>	1-10	$\epsilon$	fraction of the maximum source-panel diagonal ( $t_{max}$ ) used to form $\epsilon_{10}$ , which is the offset distance used in the calculation of $dV_t/ds_t$ . If $\epsilon = 0$ , the derivative will not be calculated.
<u>Card 13</u>	1-4	END	Control card—columns 1-4 contain the word END <sub>b</sub> , where b represents a blank.

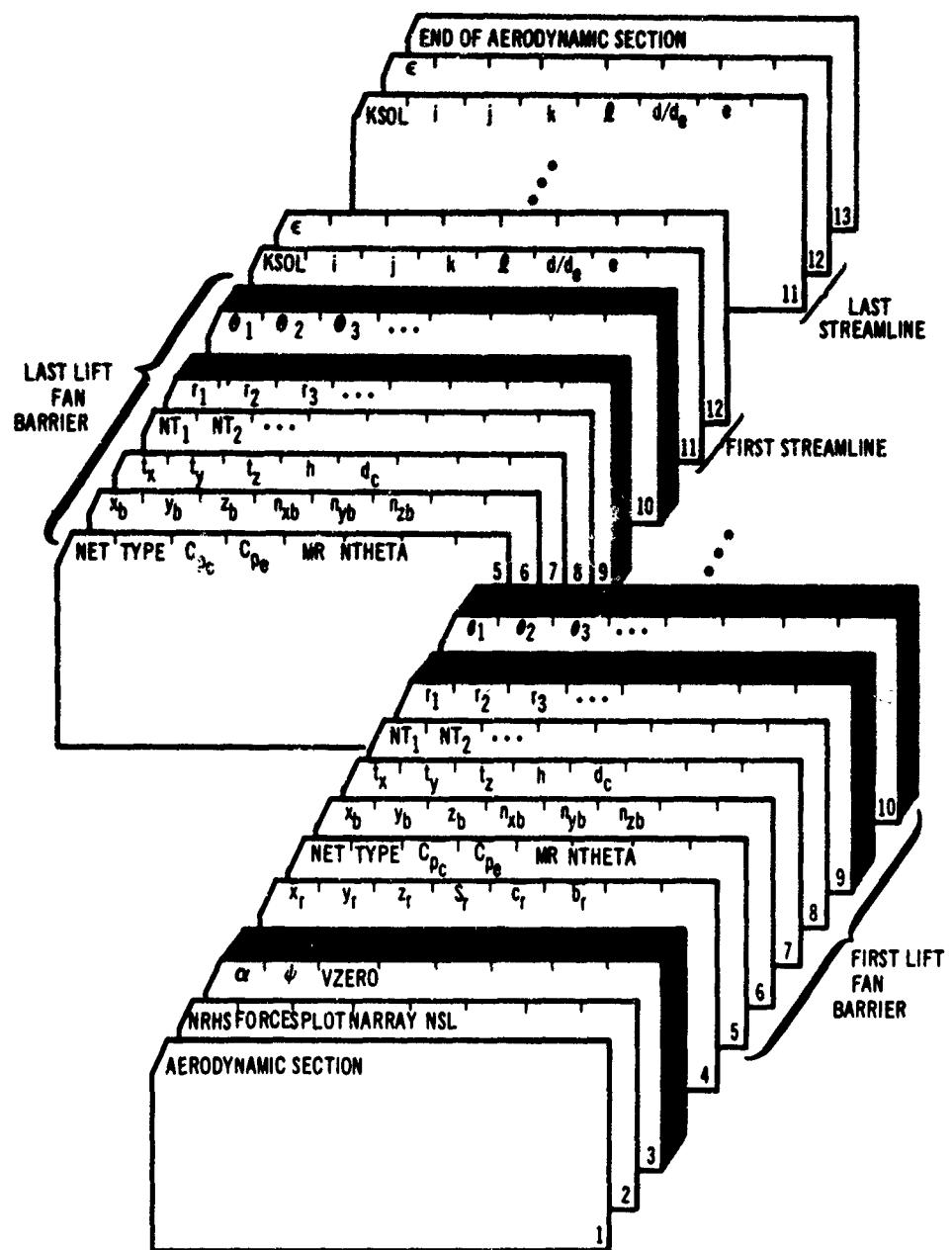


Figure 26. Data Card Arrangement for Aerodynamic Section.

### 3.2.3 Monitor Control Cards

Figure 27 displays the monitor control cards necessary to execute the potential-flow program.

#### Monitor Control Card 1

is the job card, containing the job name, priority, central processor time limit, and central memory requirement. Fields are separated by commas, the last field being terminated by a period.

- NAME is the job name, one to seven characters; the first character must be a letter. The SCOPE Operating System replaces characters 5, 6, and 7 with a unique sequence number indicating the number of jobs run through the system since dead start.
- PR is the job priority, 1 through 17 octal.
- T is the central processor time limit in seconds (octal).
- FL is the field length for the job (octal).

#### Monitor Control Card 2

is the accounting card, distinguished by the letters A, C, C, and T in columns 1 through 4. Information on this card will vary between computer installations but generally includes the user's name, location, and telephone number.

#### Monitor Control Card 3

is the compiler program call card. In the card illustrated, the RUN compiler is called into core to compile the program source decks.

- S is the compiler mode option. The letter S will cause the RUN compiler to compile and list the source decks but will not execute them.

FNAME is the file name on which the program binary routines are written. This must contain the name FNAME.

lc is the line limit (octal) on the OUTPUT file. If the line count exceeds the specified line limit, the run is terminated.

#### Monitor Control Card 4

copies all the binary routines from the INPUT file to the program file (FNAME). These binary routines are the Boeing-developed matrix routines.

INPUT is the file name on which the program binary routines are initially stored.

<u>rd</u>	is the number of binary records (each routine is one record) to be copied.
<u>Monitor Control Card 5</u>	copies one binary record (the input data) from the INPUT file to file TAPE5.
<u>Monitor Control Card 6</u>	rewinds file TAPE5.
<u>Monitor Control Card 7</u>	is the program call card. The name FNAME must appear in columns 1 through 5, followed by a period in column 6. This card loads and executes the program.
<u>Monitor Control Card 8</u>	is an end-of-record card. This card has in column 1 the punches 7-8-9.
<u>Card Set 9</u>	represents the FORTRAN source decks. Note that there is a terminal end-of-record card (monitor control card 10), but that there are no end-of-record cards between individual source decks.
<u>Monitor Control Card 10</u>	is an end-of-record card.
<u>Card Set 11</u>	represents the matrix package binary routines supplied with the program. There are end-of-record cards terminating each routine.
<u>Card Set 12</u>	represents the input data. These cards will vary depending on the number and type of configurations to be run.
<u>Monitor Control Card 13</u>	is an end-of-file card. This card has in column 1 the punches 6-7-8-9. It is used to designate the end of the job input file and must be the last card of the deck.

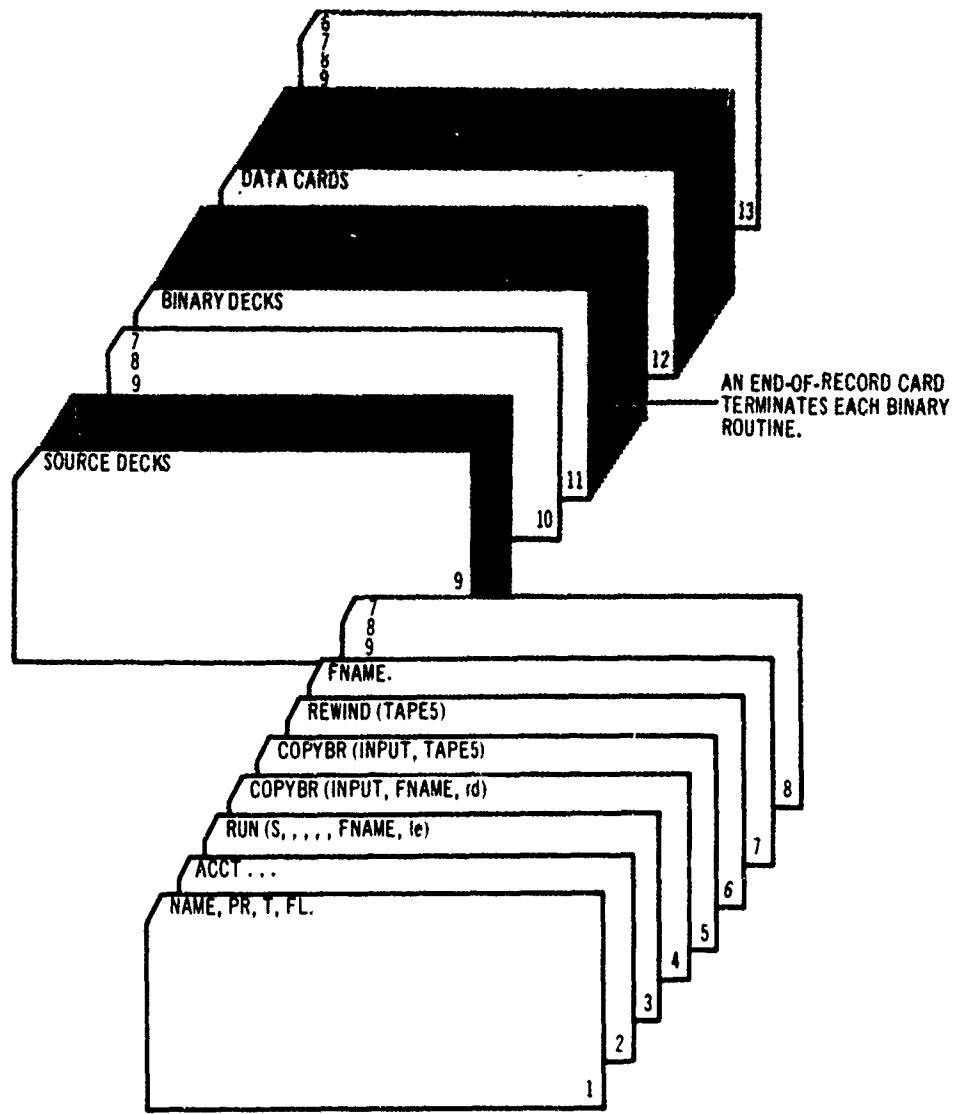


Figure 27. Monitor Control Card Arrangement for Potential-Flow Program.

### 3.2.4 Program Field Length, Timing, and Output Estimate

The maximum field length of the potential-flow program is 240,000 octal words, which is reached when the zero level or main segment and the level 1 segment containing the streamline subroutines are loaded.

Program timing is divided into central processor timing and peripheral processor timing. Central processor time is the time required to execute and control the program, while peripheral processor time is the time needed to perform all input/output functions required by the program. The central processor time is largely a function of the number of singularities needed to represent the configuration, although the number of off-body points and the number of streamlines also affect the timing. Figure 28 presents a conservative estimate of the central processor time. The peripheral processor time is generally three to eight times longer than the central processor time, depending on how the computer is loaded with other programs.

An estimate for the printed output is given by

$$\text{Lines} = 5 \cdot \text{NSING} + \text{NOFF} + 1.2 \cdot \text{NSOL} \cdot (\text{NSING} + \text{NOFF}) + 110 \cdot \text{NSL} \quad (8)$$

where NSING = number of singularities

NOFF = number of off-body points

NSOL = number of solutions desired

NSL = number of streamlines

## 3.3 BOUNDARY-LAYER PROGRAM

### 3.3.1 Machine Components

The boundary-layer program is coded in FORTRAN IV for the CDC 6600 (131k) digital computer. Control of the computer is monitored by the SCOPE Operating System. Two data files for input and output are required by the program.

### 3.3.2 Input Data Format

A description of the input format of the boundary-layer program is given in this section. The majority of the quantities in the data input are obtained from the potential-flow program. Because of the system of panels employed in the potential-flow analysis and because of the method of calculating the streamlines and parameters associated with the streamlines, it is necessary to evaluate and perhaps to interpret the potential-flow results before they are used in the boundary-layer program. Such interpretation is discussed in detail in Volume I, Section 6.

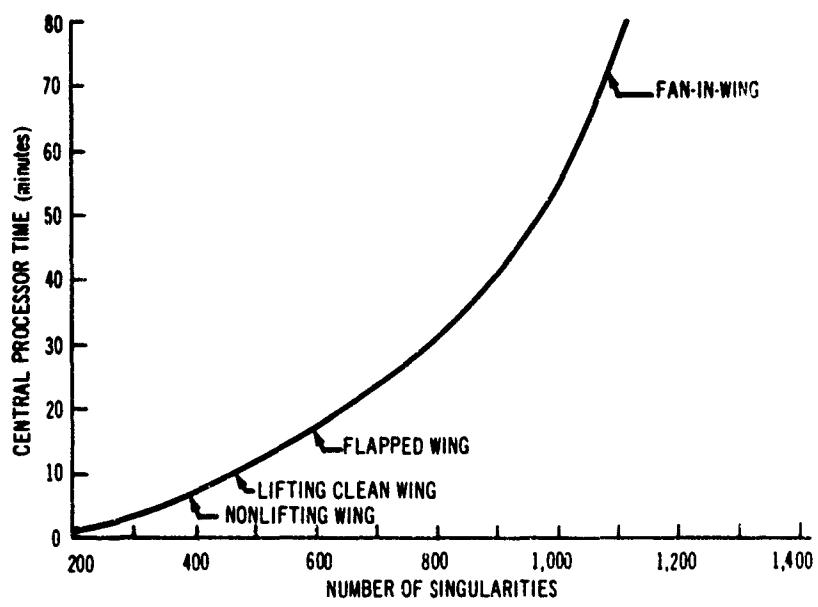


Figure 28. Potential-Flow Program Central Processor Time Estimate.

**BOUNDARY-LAYER PROGRAM CARD INPUT**

Figure 29 displays the stacked data cards for the boundary-layer program. All data, except the title card, are punched in a seven-field, ten-digit format. A decimal point is required in each data field. A description of the card input to this program follows.

	<u>Column</u>	<u>Code</u>	<u>Explanation</u>
<u>Card 1</u>	1-72	TITLE	any desired title
<u>Card 2</u>	1-10	A	potential velocity derivative at the stagnation point, or initial momentum thickness
	11-20	NX	number of points along the streamline at which the velocity is to be specified
	21-30	RINF	$R_\infty$ , Reynolds number based on the reference velocity
	31-40	STEP	= 0.; the output is printed at each integration step. ≠ 0.; the output is printed at equally spaced increments of distance, STEP, along streamline.
	41-50	SL	total length of the streamline
	51-60	FLAG	= 0.; initial value of the momentum thickness = $\frac{0.2905}{\sqrt{A * RINF}}$ ≠ 0.; initial momentum thickness = A.

The next NX cards specify the streamwise velocity and normal velocity derivative along the streamline. Each card contains the values at one point on the streamline. The points are ordered in sequence proceeding in the upstream direction along a streamline.

<u>Card Set 3</u>	1-10	s	distance from farthest downstream point along the streamline
	11-20	V	potential-flow velocity at the distance s along the streamline
	21-30	$dV_t/ds_t$	normal velocity derivative at s. The derivative of the component of velocity normal to the streamline along the surface orthogonal trajectory.

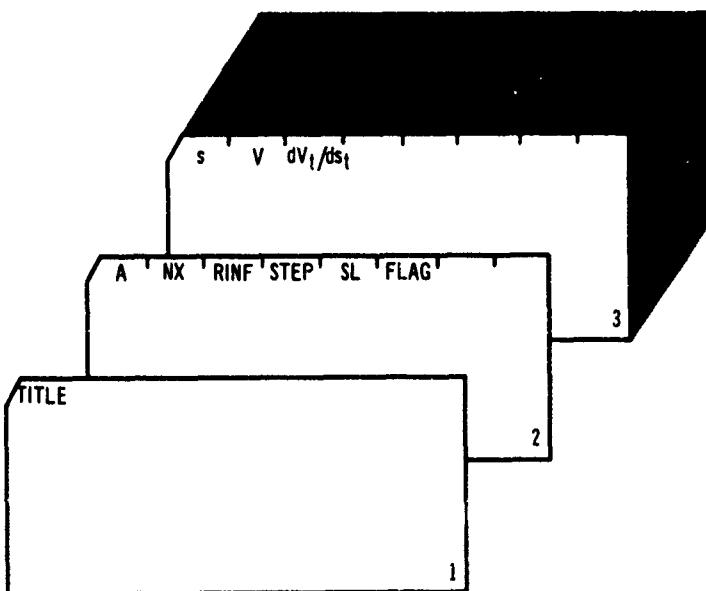


Figure 29. Data Card Arrangement for Boundary-Layer Program.

### **3.3.3 Monitor Control Cards**

Figure 30 displays the monitor control cards necessary to execute the boundary-layer program.

#### **Monitor Control Card 1**

is the job card, containing the job name, priority, central processor time limit, and central memory requirement. Fields are separated by commas, the last field being terminated by a period.

- NAME is the job name, one to seven characters; the first character must be a letter. The SCOPE Operating System replaces characters 5, 6, and 7 with a unique sequence number indicating the number of jobs run through the system since dead start.
- PR is the job priority, 1 through 17 (octal).
- T is the central processor time limit in seconds (octal).
- FL is the field length for the job (octal).

#### **Monitor Control Card 2**

is the accounting card, distinguished by the letters A, C, C, and T in columns 1 through 4. Information on this card will vary between computer installations but generally includes the user's name, location, and telephone number.

#### **Monitor Control Card 3**

is the program call card. In this case, the RUN compiler is called into core to compile and execute the source decks. The letters R, U, and N are in columns 1 through 3, followed by a period in column 4.

#### **Monitor Control Card 4**

is an end-of-record card. This card contains 7-8-9 punches in column 1.

#### **Card Set 5**

represents the source decks. Note that there is a terminal end-of-record card (monitor control card 6) but that there are no end-of-record cards between individual source decks.

#### **Monitor Control Card 6**

is an end-of-record card.

#### **Card Set 7**

represents the input data. These cards will vary, depending on the particular problem to be run.

#### **Monitor Control Card 8**

is an end-of-file card. It contains 6-7-8-9 punches in column 1. There must be only one of these per deck, and it is always the last card.

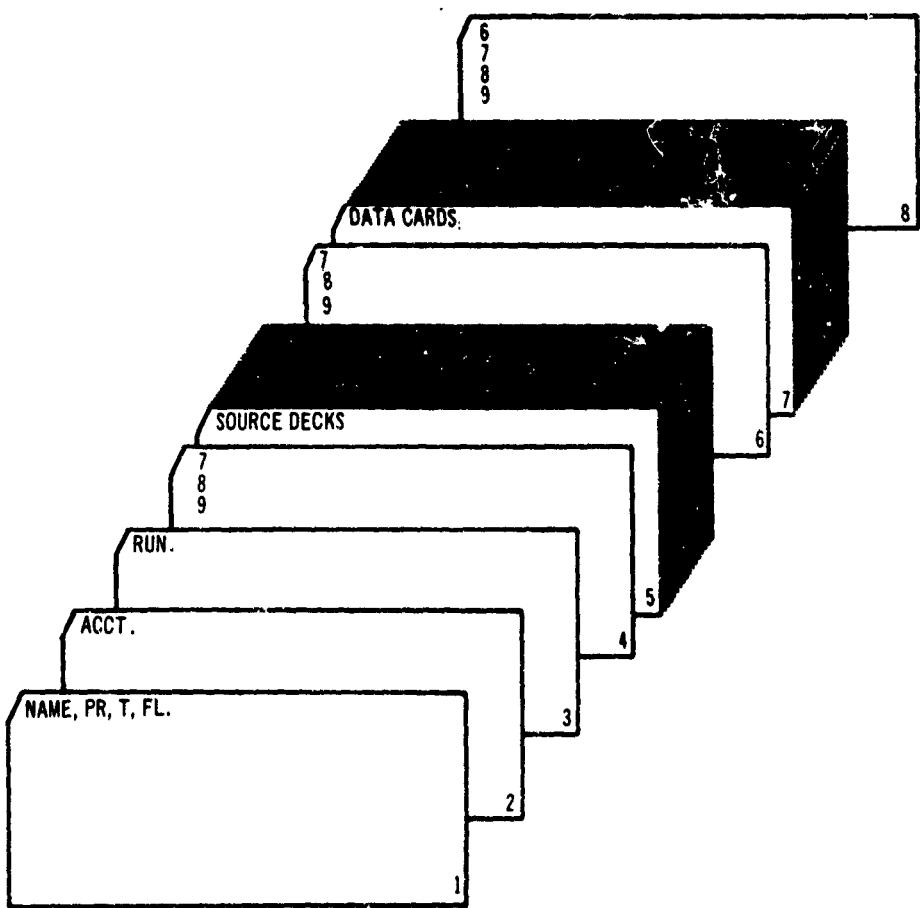


Figure 30. Monitor Control Card Arrangement for Boundary-Layer Program.

### 3.3.4 Program Field Length, Timing, and Output Estimate

A field length of 50,000<sub>8</sub> is required for compilation and execution of the boundary-layer program.

The program requires approximately 8 seconds of central processor time to process one case.

An estimate of the printed output is given by

$$\text{lines} = (N + 1) + R \quad (9)$$

where  $N$  = number of points at which the velocity is specified if STEP  $\neq 0$

$R = SL/\text{STEP}$ , the streamline length divided by the increment of spacing. If STEP = 0, R is set to 1000.

#### 4. REFERENCES

1. Rubbert, P. E., Saaris, G. R., Scholey, M. B., Standen, N. M., and Wallace, R. E., "A General Method for Determining the Aerodynamic Characteristics of Fan-in-Wing Configurations: Volume I—Theory and Application," The Boeing Company, USAVLABS Technical Report 67-61A, U.S. Army Aviation Materiel Laboratories, Fort Eustis, Virginia, 1967.
2. "Control Data 6400/6600 Computer Systems SCOPE Reference Manual," Control Data Corporation, Publication No. 60173800, Palo Alto, California, September 1966.
3. "FORTRAN Reference Manual," Control Data Corporation, Publication No. 60174900, Palo Alto, California, October 1966.
4. "ASCENT/ASPER Reference Manual," Control Data Corporation, Publication No. 60172700, Palo Alto, California, July 1966.

## APPENDIX I

### FLOW CHARTS

A problem flow chart, supplemented by flow charts of the three programs, is presented in this appendix. (See Figures 31 through 43.)

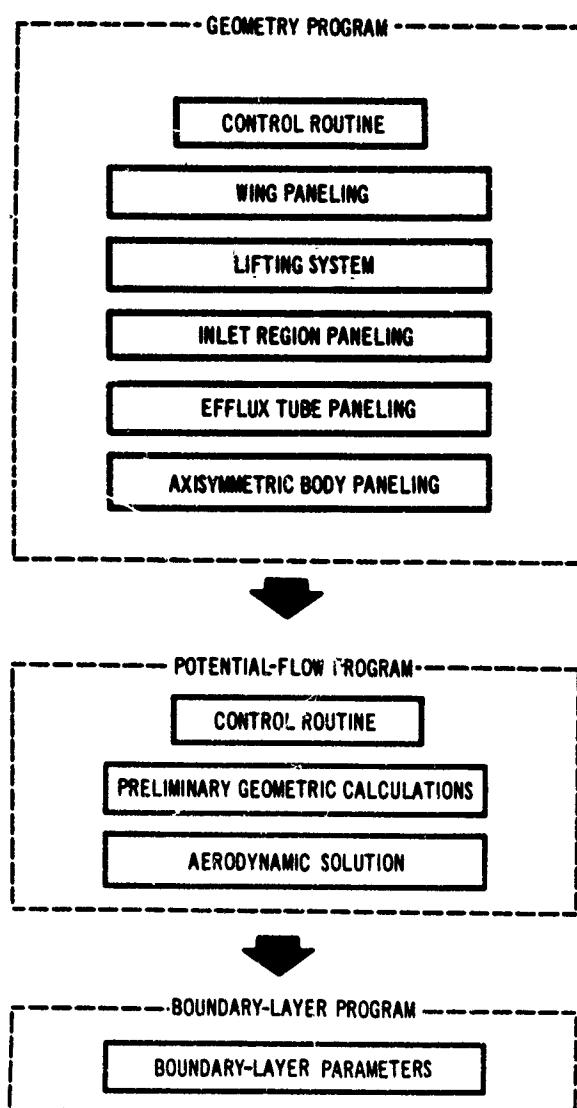


Figure 31. Fan-in-Wing Problem Flow Chart.

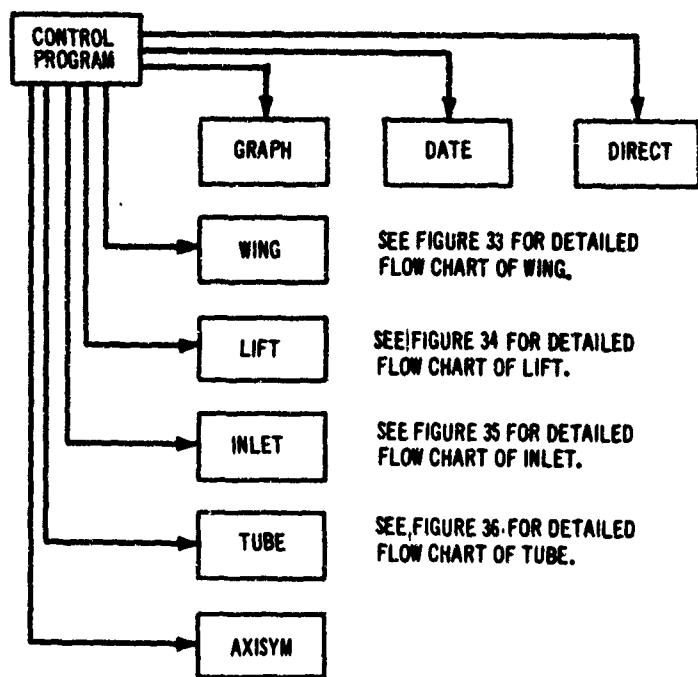
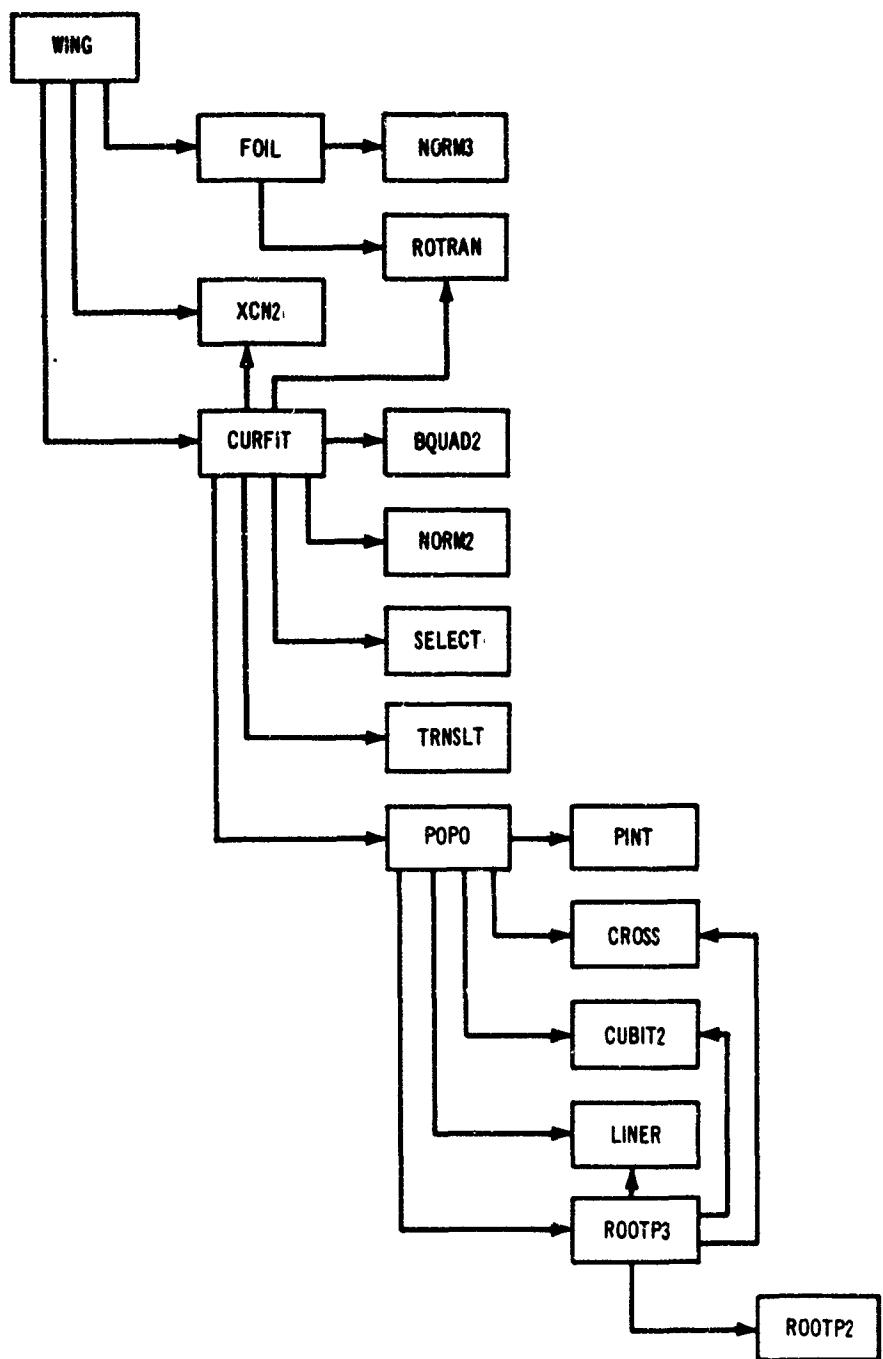


Figure 32. Geometry Program Flow Chart.



**Figure 33. Geometry Program—Subroutine WING (Wing Paneling).**

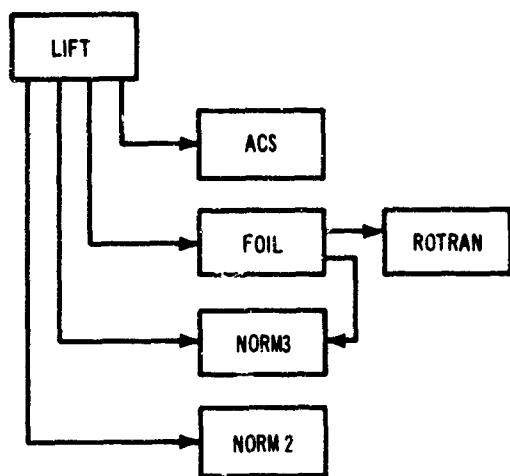


Figure 34. Geometry Program—Subroutine LIFT (Lifting System).

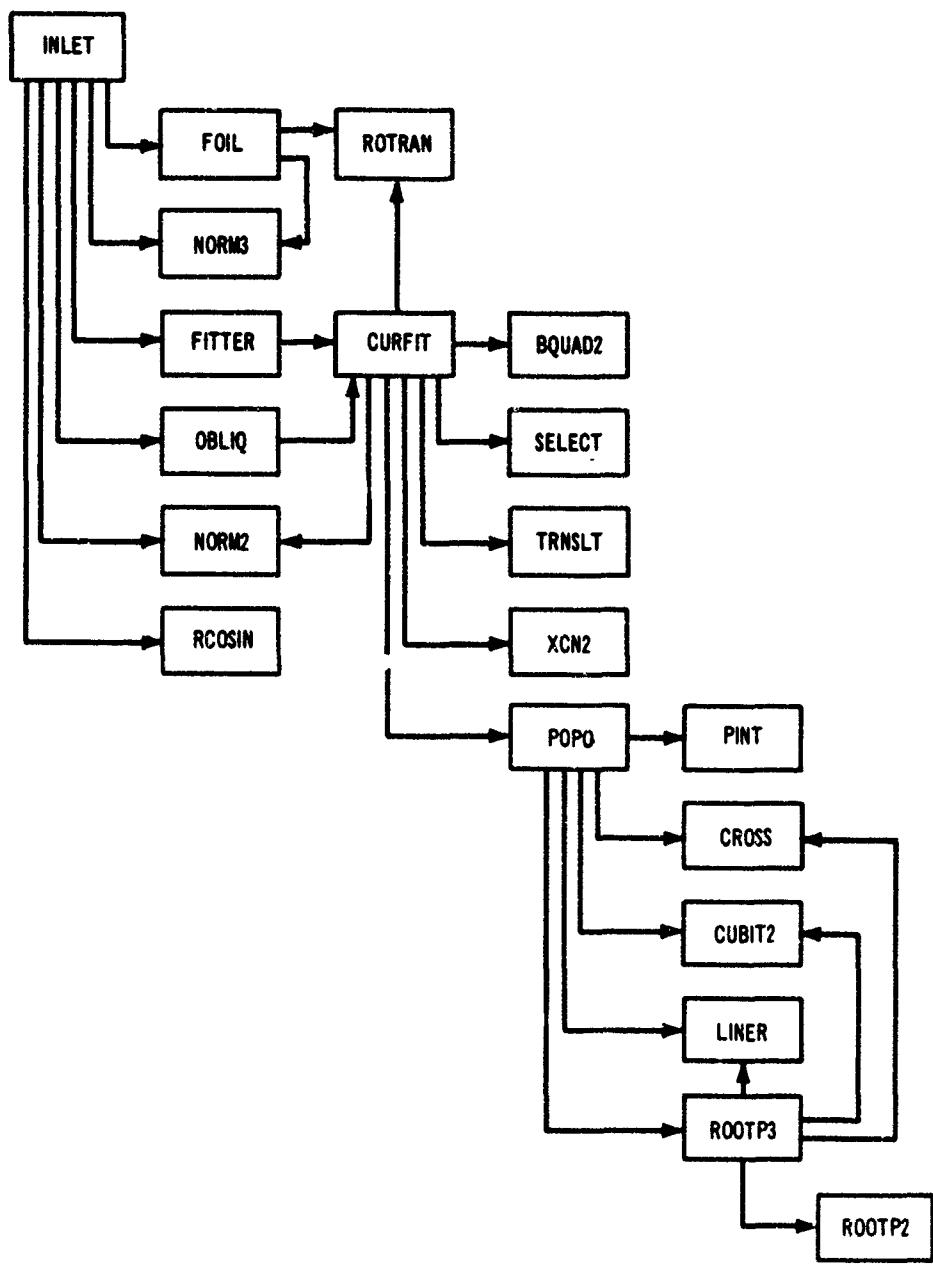
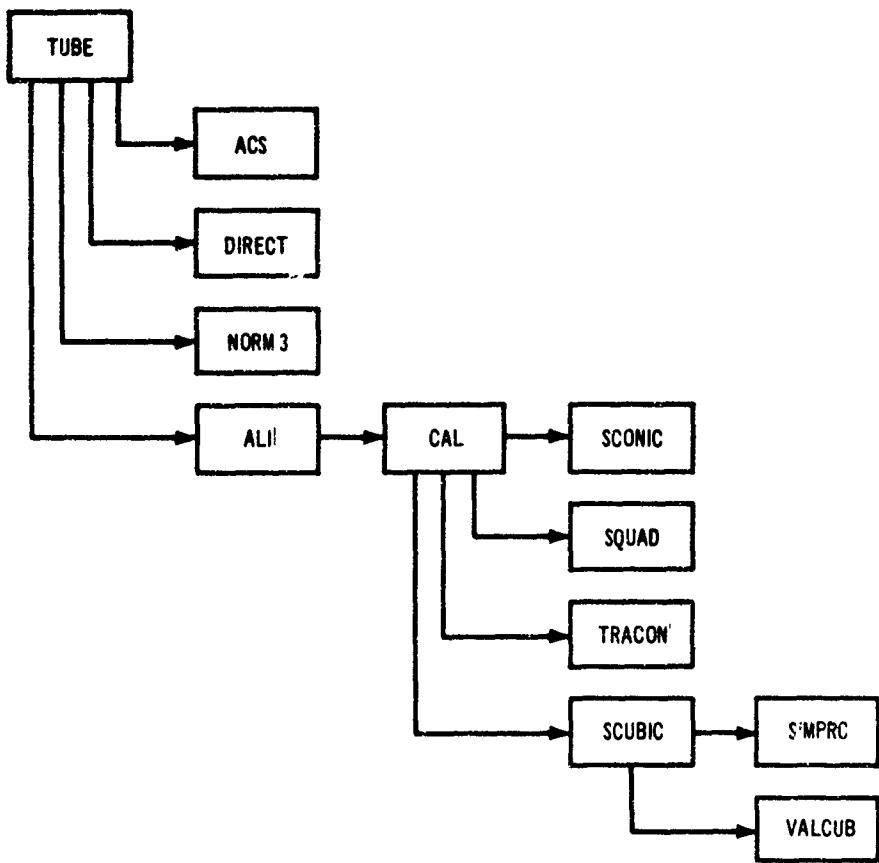
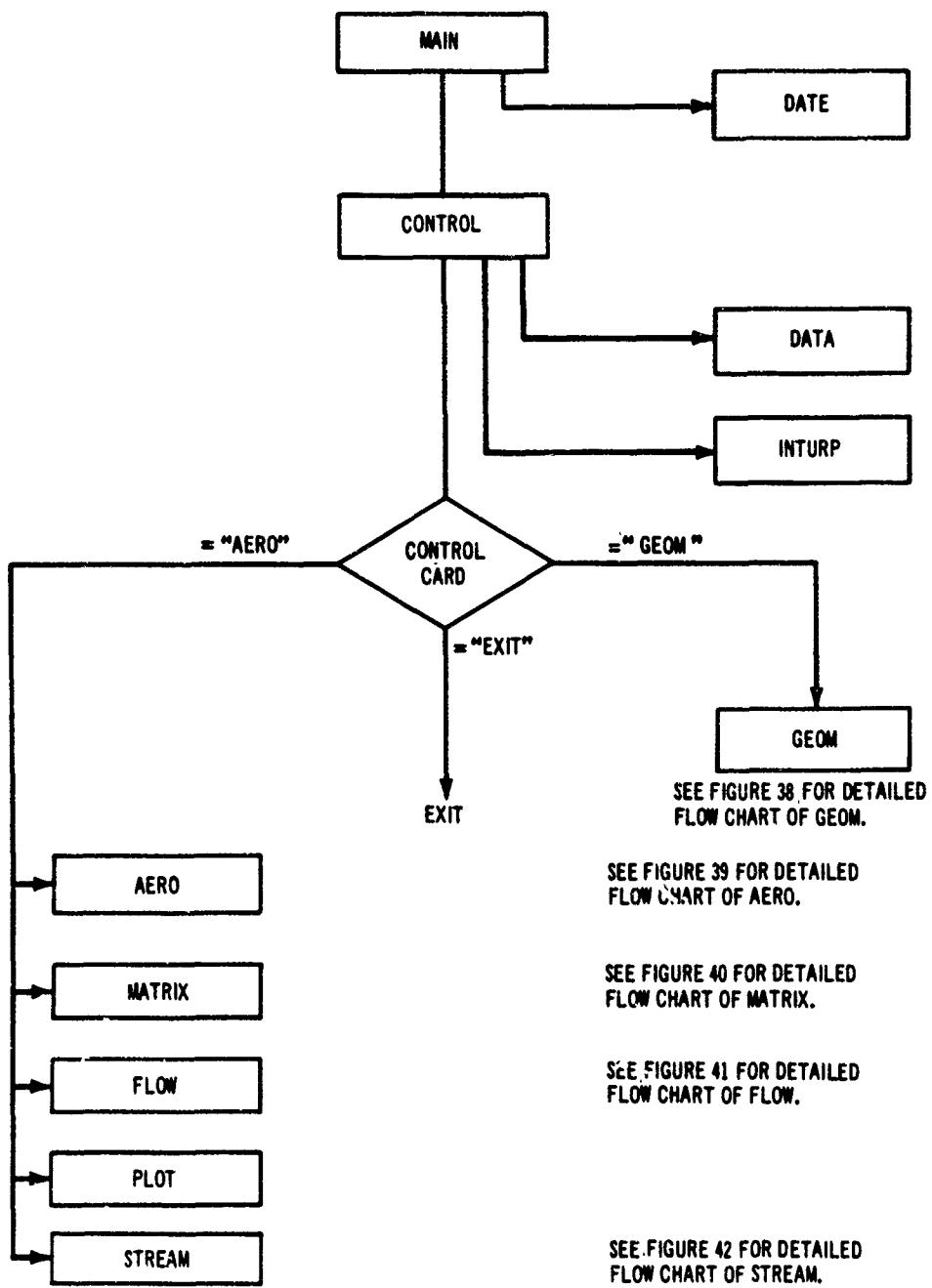


Figure 35. Geometry Program—Subroutine INLET (Inlet Region Paneling).



**Figure 36. Geometry Program—Subroutine TUBE (Efflux Tube Paneling).**



**Figure 37. Potential-Flow Program Flow Chart.**

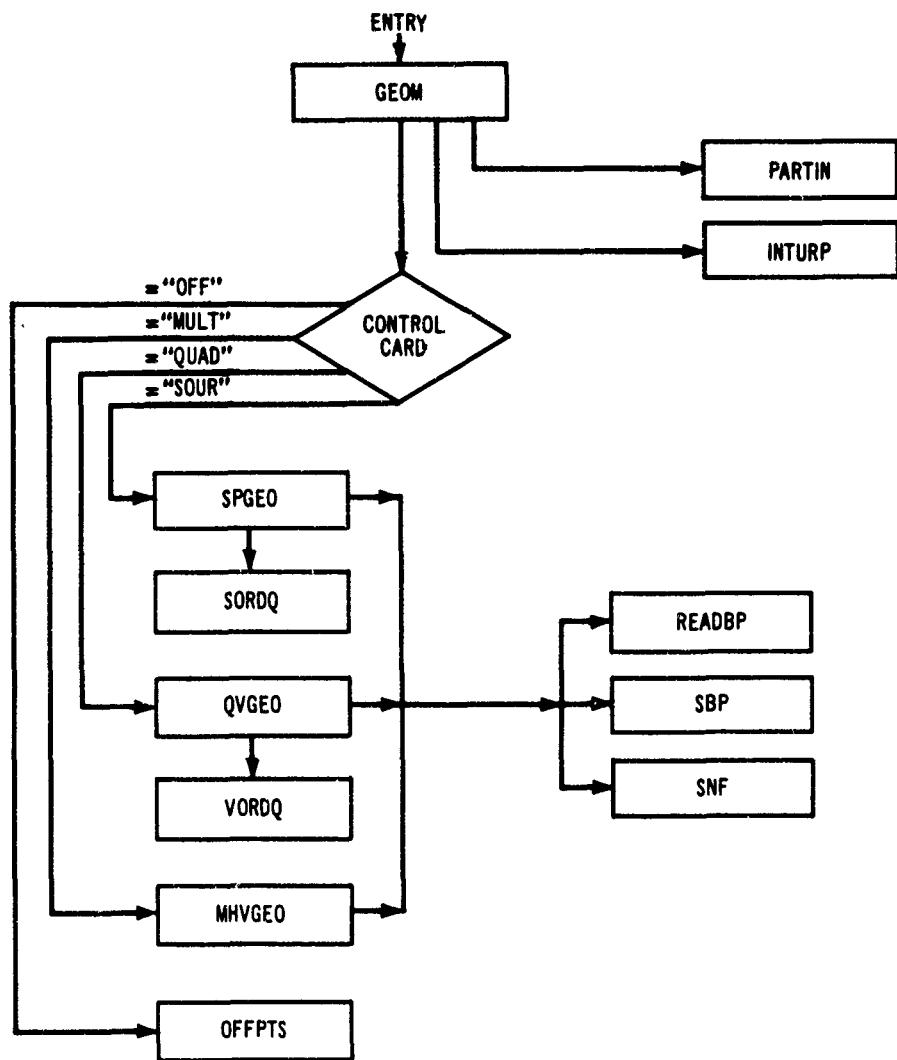
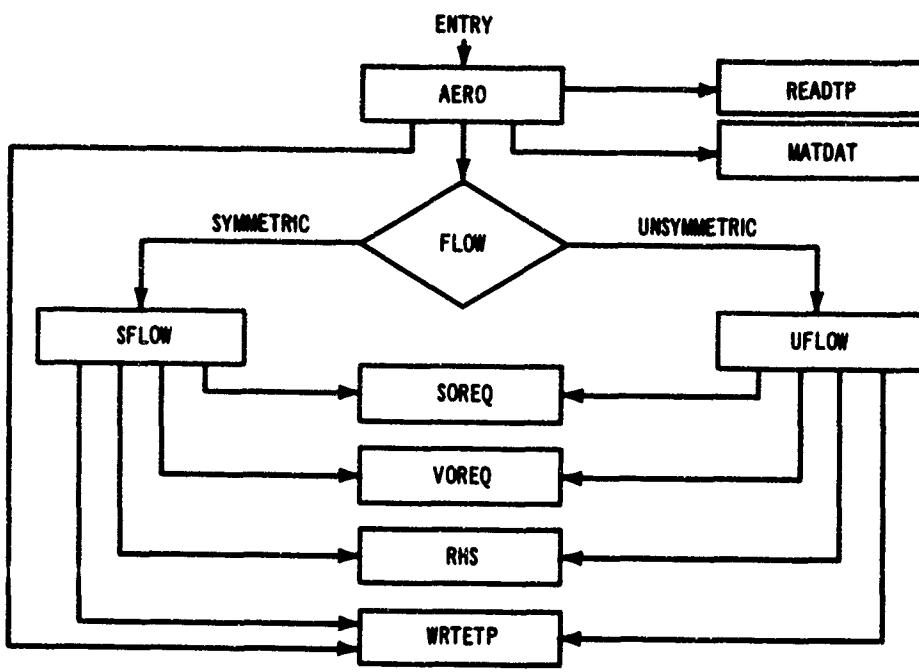
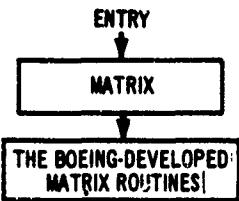


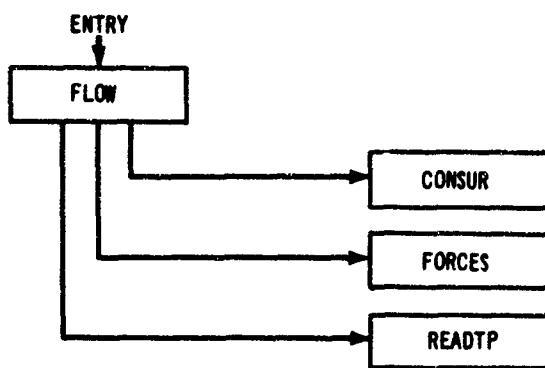
Figure 38. Potential-Flow Program—Subroutine GEOM (Preliminary Geometry).



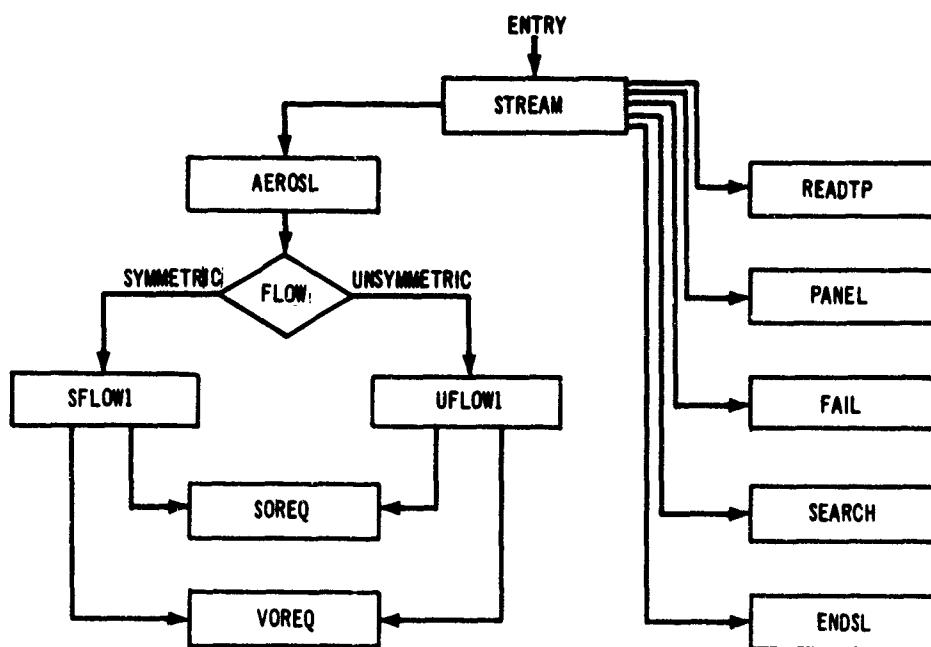
**Figure 39. Potential-Flow Program—Subroutine AERO  
(Velocity Matrix Formation).**



**Figure 40. Potential-Flow Program—Subroutine MATRIX  
(Matrix Equation Solution).**



**Figure 41.** Potential-Flow Program—Subroutine FLOW  
(Potential-Flow Solution).



**Figure 42.** Potential-Flow Program—Subroutine STREAM  
(Streamline Trajectories).

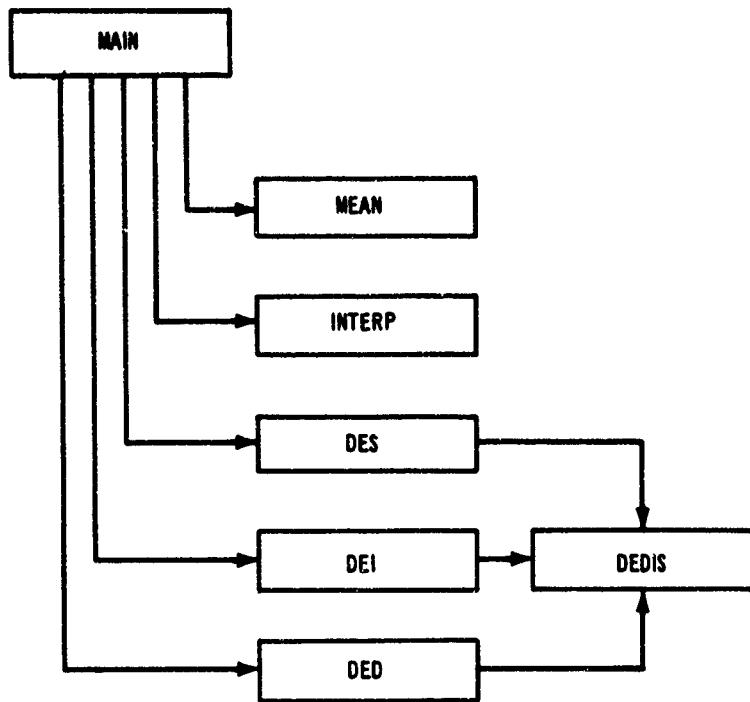


Figure 43. Boundary-Layer Program Flow Chart.

## APPENDIX II

### SEGMENTATION STRUCTURE

The geometry and potential-flow programs exceed the capacity of a single core load and utilize the segmentation feature of the loader. Figures 44 and 45 illustrate the segmentation structure and list the subroutines required for each segment, respectively.

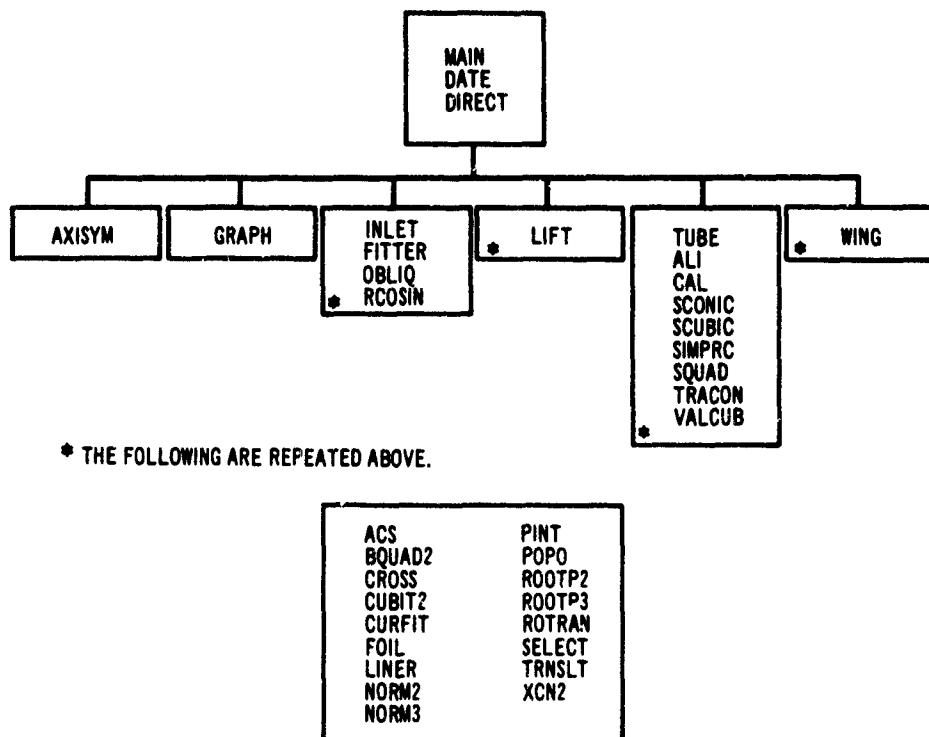
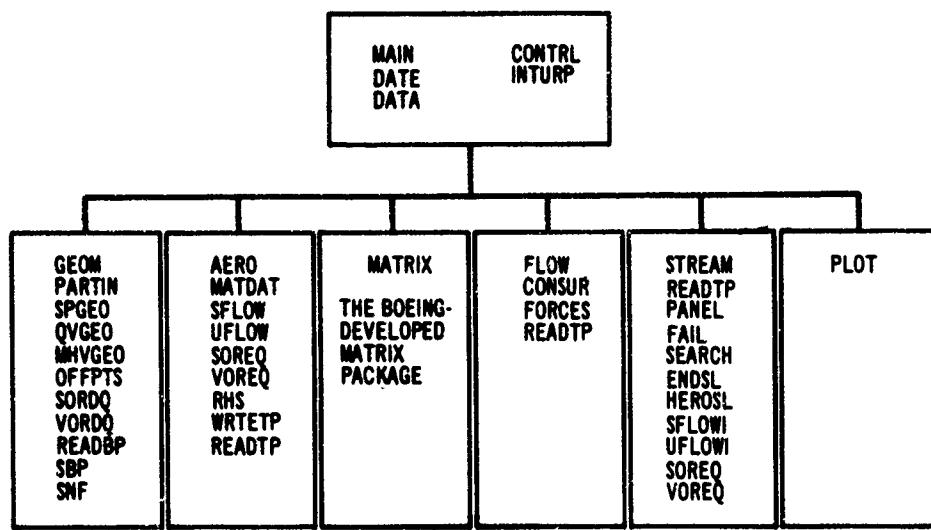


Figure 44. Geometry Program Segmentation Structure.



**Figure 45. Potential-Flow Program Segmentation Structure.**

## APPENDIX III

### SUBROUTINE DESCRIPTIONS

The subroutines of the three programs—geometry program, potential-flow program, and boundary-layer program—are described in this appendix. The purpose, method, usage, subprograms called, error returns, and restrictions are given in detail for each subroutine.

A. Geometry program subroutine descriptions.—The subroutines listed alphabetically below are discussed in this section.

Subroutine Index					
<u>Subroutine</u>	<u>Page</u>	<u>Subroutine</u>	<u>Page</u>	<u>Subroutine</u>	<u>Page</u>
ACS	98	INLET	114	ROTRAN	136
ALI	99	LIFT	119	SCONIC	137
AXISYM	100	LINER	121	SCUBIC	138
BQUAD2	102	MAIN	122	SELECT	139
CAL	103	NORM2	123	SIMPRC	141
CROSS	104	NORM3	124	SQUAD	143
CUBIT2	106	OBLIQ	125	TRACON	144
CURFIT	107	PINT	128	TRNSLT	145
DATE	109	POPO	129	TUBE	146
DIRECT	110	RCOSIN	131	VALCUB	149
FITTER	111	ROOTP2	133	WING	150
FOIL	112	ROOTP3	134	XCN2	152
GRAPH	113				

**SUBJECT:** FORTRAN IV Subroutine ACS

**PURPOSE:** To find the x, y, and z coordinates of the average center of four points and the direction cosines of the normal to the surface at the average center

**METHOD:** The diagonal vectors  $\vec{T}_1$  and  $\vec{T}_2$ , shown in Figure 46, are found from the input coordinates. Their cross product,  $\vec{T}_2 \times \vec{T}_1$ , is calculated, from which the direction cosines of the normal are found. The coordinates of the average center are found by averaging the coordinates of the four input points.

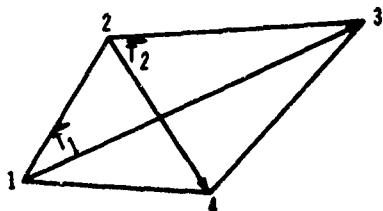


Figure 46. Subroutine ACS — Panel Diagonals.

**USAGE:** CALL ACS (X, Y, Z, OUT)

DIMENSION X(4), Y(4), Z(4), OUT (6)

**Input:**  $\begin{matrix} X \\ Y \\ Z \end{matrix}$  } = arrays of the x, y, and z coordinates of the four points

**Output:** OUT (1) = x coordinate of the average center  
 OUT (2) = y coordinate of the average center  
 OUT (3) = z coordinate of the average center  
 OUT (4) = direction cosine of the normal with respect to the x-axis  
 OUT (5) = direction cosine of the normal with respect to the y-axis  
 OUT (6) = direction cosine of the normal with respect to the z-axis

**SUBPROGRAMS CALLED:** None

**ERROR RETURNS:** None

**RESTRICTIONS:** None

**SUBJECT:** FORTRAN IV Function ALI  
**PURPOSE:** Given a curve defined by a cubic and two of the three quantities a, b, and s, to find the third quantity, where s is the length of the cubic from x = a to x = b  
**METHOD:** Subroutine CAL is used to compute the arc length of the cubic over any interval.

$$S = \int_a^b \sqrt{1 + \left(\frac{dy}{dx}\right)^2} dx \quad (10)$$

If a or b is to be calculated, an iterative technique is used.

**USAGE:** J = ALI (C, A, B, S, K)

**DIMENSION C(8)**

**Input:**  
 C(1) = 1.0  
 C(2) = minimum x for which cubic is defined  
 C(3) = maximum x for which cubic is defined  
 C(4) = 3.0  
 C(5) =  $c_0$   
 C(6) =  $c_1$   
 C(7) =  $c_2$   
 C(8) =  $c_3$  } coefficients of the cubic equation  

$$y = c_0 + c_1x + c_2x^2 + c_3x^3$$

A = a, left end of curve segment  
 B = b, right end of curve segment  
 S = s, arc length of curve segment  
 K > 0, given a and s, find b  
 = 0, given a and b, find s  
 < 0, given b and s, find a

**Output:** a, b, or s, depending on the value of code K

J = 1, if C(4) is not 3.0  
 = 5,000, if cubic not defined at a  
 = 10,000, if cubic not defined at b  
 < 0, M, error return to M from subroutine  
 CAL  
 = 0, success

**SUBPROGRAMS CALLED:** CAL  
LOOK

**ERROR RETURNS:** J, see USAGE

**RESTRICTIONS:** None

**SUBJECT:** FORTRAN IV Subroutine AXISYM  
**PURPOSE:** To find the x, y, and z coordinates of an axisymmetric body  
**METHOD:** This subroutine accepts input data in an  $(r, z, \theta)$  coordinate system as shown in Figure 47, and transforms it into an  $(x, y, z)$  system.

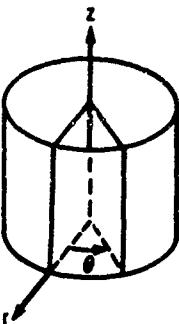


Figure 47. Subroutine AXISYM—Polar Coordinate System.

Input consists of a table of  $n$   $\theta$ 's and either one or  $n$  tables of  $r$  versus  $z$ , allowing a separate  $r-z$  table for each  $\theta$  if desired.

The  $x$ ,  $y$ , and  $z$  coordinates calculated from these tables may be rotated and distorted, and then translated to any position before or after distortion. The following equations define the several transformations performed on the input  $(r, z, \theta)$  coordinates:

Transformation to  $(x, y, z)$  system:

$$\begin{aligned} x_1 &= r \cos\theta \\ y_1 &= r \sin\theta \\ z_1 &= z \end{aligned} \quad (11)$$

Rotation:

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \quad (12)$$

First Translation:

$$\begin{bmatrix} x_3 \\ y_3 \\ z_3 \end{bmatrix} = \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} + \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} \quad (13)$$

**Distortion:**

$$\begin{bmatrix} x_4 \\ y_4 \\ z_4 \end{bmatrix} = \begin{bmatrix} k_1 x_3 \\ k_2 y_3 \\ k_3 z_3 \end{bmatrix} \quad (14)$$

**Last Translation:**

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_4 \\ y_4 \\ z_4 \end{bmatrix} + \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix} \quad (15)$$

The variables used in the above equations are defined as follows:

$r$ ,  $\theta$ , and  $z$  are body-defining coordinates.

$a_{11}$ ,  $a_{12}$ , and  $a_{13}$  are direction cosines of the  $x_2$ -axis with respect to the  $(x_1, y_1, z_1)$  system.

$a_{21}$ ,  $a_{22}$ , and  $a_{23}$  are direction cosines of the  $y_2$ -axis with respect to the  $(x_1, y_1, z_1)$  system.

$a_{31}$ ,  $a_{32}$ , and  $a_{33}$  are direction cosines of the  $z_2$ -axis with respect to the  $(x_1, y_1, z_1)$  system.

$x_0$ ,  $y_0$ , and  $z_0$  are origin coordinates of the  $(x_2, y_2, z_2)$  system with respect to the  $(x_3, y_3, z_3)$  system.

$k_1$ ,  $k_2$ , and  $k_3$  are distortion factors for the  $(x_3, y_3, z_3)$  system.

$x_p$ ,  $y_p$ , and  $z_p$  are origin coordinates of the  $(x_4, y_4, z_4)$  system with respect to the  $(x, y, z)$  system.

**USAGE:**

CALL AXISYM

Input for subroutine AXISYM is detailed in Section 3.1.2.

Output consists of  $x$ ,  $y$ , and  $z$  coordinates of the body, optionally punched on cards.

**SUBPROGRAMS  
CALLED:**

None

**ERROR RETURNS:**

None

**RESTRICTIONS:**

None

**SUBJECT:** FORTRAN IV Subroutine BQUAD2

**PURPOSE:** To find coefficients of a cubic or either of two quadratics passing through all or three of four given points

**METHOD:** Subroutine BQUAD2 calculates three quadratic coefficients or four cubic coefficients, depending on the value of the input K.

If K = 1, the coefficients of the quadratic  $Q_1(x)$  passing through the first three points are calculated.

If K = 3,  $Q_2(x)$  is calculated, passing through the last three input points.

When K = 2, a cubic C(x) is calculated using Equation 16.

$$\begin{aligned}
 C(x) &= \frac{(x_3 - x) Q_1(x) + (x - x_2) Q_2(x)}{x_3 - x_2} \\
 &= c_0 + c_1 x + c_2 x^2 + c_3 x^3
 \end{aligned} \quad (16)$$

**USAGE:** CALL BQUAD2 (X, Y, C, K, J)  
 DIMENSION C(4), X(4), Y(4)

**Input:** X } = array of x and y coordinates of the four  
 Y } input points

**K** = 1,  
 = 2, see METHOD  
 = 3,

**Output:** C = array of quadratic or cubic coefficients, depending on K

**J** = 1, success  
 = 2, division by zero was attempted

**SUBPROGRAMS CALLED:** None

**ERROR RETURNS:** J, see USAGE

**RESTRICTIONS:** The four input x values must be strictly monotonically increasing.

**SUBJECT:** FORTRAN IV Subroutine CAL

**PURPOSE:** To calculate the arc length  $s$  of a cubic over a specified interval, from  $x = a$  to  $x = b$

**METHOD:** Subroutine CAL calls subroutine SCUBIC to find the arc length of the cubic.

**USAGE:** CALL CAL (C, X, S, J)  
DIMENSION C(5), X(2)

Input: C(1) = 3.0  
C(2) =  $c_0$   
C(3) =  $c_1$   
C(4) =  $c_2$   
C(5) =  $c_3$       coefficients of the cubic equation  
$$y = c_0 + c_1x + c_2x^2 + c_3x^3$$

X(1) = a or b  
X(2) = b or a

Output: S =  $s$ , the arc length  
J = 0, success  
= -1, C(1)  $\neq$  3.0  
= -2, error return from SCUBIC

**SUBPROGRAMS CALLED:** SCONIC  
SCUBIC  
SQUAD  
TRACON

**ERROR RETURNS:** J, see USAGE

**RESTRICTIONS:** None

SUBJECT:

**FORTRAN IV Subroutine CROSS**

PURPOSE:

To find an intersection within a specified interval [a, b] of two functions given a first approximation, x, to the abscissa of the intersection

METHOD:

Let  $\Phi_r$  and  $\Phi_s$  be the two functions (evaluated by subroutines DUMR and DUMS, respectively). Subroutine CROSS finds two abscissas in the specified interval,  $x_p$  and  $x_N$ , such that the difference function,  $g(x) = \Phi_r - \Phi_s$  changes sign between them (see Figure 48).

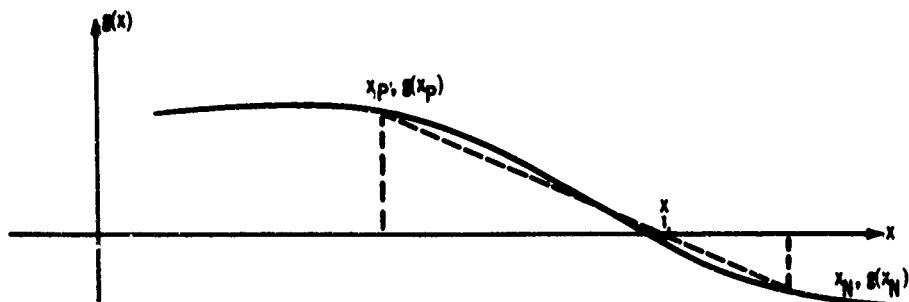


Figure 48. Subroutine CROSS—Intersection of Two Functions.

A new value of x is calculated using these two points from the formula

$$x = x_N - \left[ \frac{x_p - x_N}{g(x_p) - g(x_N)} \right] \cdot g(x_N) \quad (17)$$

The difference function is calculated at this x value. If  $g(x)$  has the same sign as  $g(x_p)$ , x replaces  $x_p$ . Otherwise, x replaces  $x_N$ . This process is repeated until successive x values are equal,  $g(x) = 0$ , or NI (see USAGE) iterations have been made.

USAGE:

CALL CROSS (DUMR, CR, DUMS, CS, A, B, NI, X, Y, J)

DIMENSION CR(m), CS(n)

Note: m and n are large enough to agree with DUMR and DUMS, respectively.

EXTERNAL DUMR, DUMS

Input: DUMR } names of subroutines that evaluate the  
DUMS } functions  $\Phi_r$  and  $\Phi_s$

$\left. \begin{array}{l} CR \\ CS \end{array} \right\}$  arrays of coefficients that define  
functions; dimensioned to agree with  
DUMR and DUMS

$\left. \begin{array}{l} A \\ B \end{array} \right\}$  = [a, b], interval to be searched for  
intersection

NI = maximum number of iterations

X = first approximation to the abscissa of  
the intersection

Output: NI = number of iterations required to find the  
intersection

X = abscissa of the intersection

Y = ordinate of the intersection

J = 0, success  
= 1, no intersection found  
= -1, error in DUMR  
= -2, error in DUMS  
= -3, NI iterations exceeded

**SUBPROGRAMS**

**CALLED:**

DUMR and DUMS. These subroutines must have a  
calling sequence of the form

CALL DUMR (C, X, Y, J),

where inputs are C, an array containing the function  
coefficients, and X, the value of the independent  
variable. Outputs are Y, the function value at X, and  
J, an error return that equals zero on success.

**ERROR RETURNS:** J, see USAGE

**RESTRICTIONS:** A < B

**SUBJECT:** FORTRAN IV Subroutine CUBIT2

**PURPOSE:** To evaluate a cubic and its first two derivatives at a given value of the independent variable

**METHOD:** Let  $y = c_0 + c_1x + c_2x^2 + c_3x^3$  represent the cubic and define

$$A = c_2 + c_3x, \quad (18)$$

$$B = c_1 + Ax. \quad (19)$$

Then

$$Y = c_0 + Bx, \quad (20)$$

$$\frac{dy}{dx} = B + x(A + c_3x), \quad (21)$$

$$\frac{d^2y}{dx^2} = 2c_2 + 6c_3x \quad (22)$$

**USAGE:** CALL CUBIT2 (C, X, Y, J)  
**DIMENSION** C(4), Y(4)

**Input:** C(1) =  $c_0$   
 C(2) =  $c_1$   
 C(3) =  $c_2$   
 C(4) =  $c_3$

X = given value of the independent variable

**Output:** Y(1) = function value at x  
 Y(2) = 1, 0  
 Y(3) = first derivative at x  
 Y(4) = second derivative at x

J = 0

**SUBPROGRAMS CALLED:** None

**ERROR RETURNS:** None

**RESTRICTIONS:** None

SUBJECT:

**FORTRAN IV Subroutine CURFIT**

PURPOSE:

To find the z coordinate corresponding to a given x,  
given a table of x versus z

METHOD:

The input x-z table defines an airfoil, as shown in  
Figure 49.

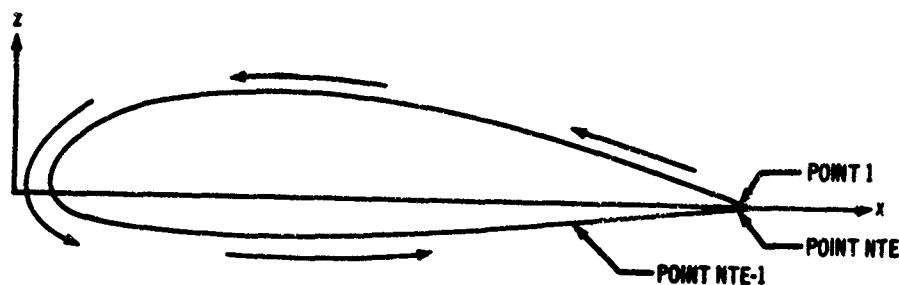


Figure 49. Subroutine CURFIT—Airfoil Definition.

An input code indicates whether the z coordinate is to be evaluated on the upper or lower surface. The specified surface definition is then searched to find the interval in which the given x lies. After a transformation to prevent a double value, subroutine BQUAD2 is called to calculate the coefficients of a biquadratic passing through the four adjacent points. Subroutine POPO calculates the intersection between this biquadratic and the vertical line passing through the given x. The coordinates of this intersection are then transformed back to the original coordinate system, and control is returned to the calling program.

USAGE:

CALL CURFIT (X, Z, NLE, NTE, XIN, CODE, ZOUT)

DIMENSION X(n), Z(n) ( $200 \geq n \geq NTE$ )

Input:       $X_j$  = arrays containing x and z coordinates  
                 $Z_j$  = defining the airfoil

NLE = subscript of leading-edge point in the  
x and z defining arrays

NTE = subscript of trailing-edge point in the  
x and z defining arrays. This number  
is also the number of points defining  
the airfoil, since the last point in the  
array is required to be the trailing edge.

XIN = the x value at which the z coordinate is  
desired

CODE  $\geq 0.$ , if upper surface z coordinate is desired

< 0., if lower surface z coordinate is desired

Output: ZOUT = z coordinate on the desired surface corresponding to x = XIN

**SUBPROGRAMS  
CALLED:**

BQUAD2  
NORM2  
POPO  
ROTRAN  
SELECT  
TRNSLT  
XCN2

**ERROR RETURNS:**

Subroutine CURFIT will return unsuccessfully in two cases: first, if the input x value is not within the range of the defining table, CURFIT returns after printing its input arguments; second, when an error flag is returned from subroutine POPO.

If subroutine BQUAD2 returns an error flag, a comment to that effect is printed, but the program attempts to continue processing.

**RESTRICTIONS:**

The defining x and z coordinates must be ordered as described in the METHOD; that is, the x values must be strictly monotonically decreasing from the upper trailing edge forward to the leading edge, and strictly monotonically increasing back to the lower trailing edge.

Note: The two trailing-edge points need not be coincident, nor does the leading edge need be defined at x = 0. The upper trailing edge is simply the first point in the table, the leading edge is that point with the minimum x coordinate, and the lower trailing edge is the last point in the table.

**SUBJECT:** FORTRAN IV Subroutine DATE

**PURPOSE:** To obtain the current date

**METHOD:** This is a dummy subroutine that returns blanks. An appropriate change may be made at the user's installation to return the true date.

**USAGE:** CALL DATE (A, B)

Output: A = first portion of the date (alphabetic)  
B = last portion of the date (alphabetic)

**SUBPROGRAMS  
CALLED:** None

**ERROR RETURNS:** None

**RESTRICTIONS:** None

**SUBJECT:** FORTRAN IV Function DIRECT

**PURPOSE:** To search a directory of words for a given word or to produce a word for a given subscript

**METHOD:** When  $K \leq 0$  (see USAGE), function DIRECT searches through a table of 40 BCD words for the given word. If a match is found, then DIRECT returns the subscript of that word in its table. If the given word is not in the table, then the function value is set to -1.

If a word corresponding to a given subscript is desired, then the function value is set to that word. A check is made to see whether the subscript is larger than 40; if so, the function value is set to 999999 (BCD).

**USAGE:**  $I = \text{DIRECT} (\text{WORD}, K)$

Input: WORD = the hollerith word for which a subscript is desired if  $K \leq 0$ ; ignored when  $K > 0$ .

$K \leq 0$ , indicates that the subscript corresponding to WORD is desired  
 $> 0$ , indicates that the word with subscript K is desired

Output: I = either a subscript or a hollerith word, depending on K. Care must be taken to ensure that this argument is in the desired mode, or the FORTRAN compiler will automatically change the mode across the equal sign.

**SUBPROGRAMS CALLED:** None

**ERROR RETURNS:** I = -1, given word not in the directory  
I = 6H999999, K greater than 40

**RESTRICTIONS:** K  $\leq 40$

**SUBJECT:** FORTRAN IV Subroutine FITTER

**PURPOSE:** Given two airfoils defined by a table of x/c and z/c values, this subroutine redefines one of the airfoils so that both airfoils are specified at the same x/c values.

**METHOD:** The longer of the two tables is used as the base table, the shorter being matched to it. If both tables are the same length, the first table is used as the base table.  
After checking which table to use as the base table, the two tables are matched using subroutine CURFIT.

**USAGE:** CALL FITTER (XC, ZC, N)  
DIMENSION XC(200,2), ZC(200,2), N(2)

**Input:**

XC(i, 1)	= airfoil-defining x/c and z/c values
ZC(i, 1)	= for first section, i = 1, N(1)
XC(i, 2)	= airfoil x/c and z/c values for second
ZC(i, 2)	= section, i = 1, N(2)
N(1)	= number of defining points of the first airfoil section
N(2)	= number of defining points of the second airfoil section

**Output:** Either the first or second airfoil section will be altered so that the x/c values correspond to those in the other section. The criterion for deciding which section to alter is described in METHOD.

**SUBPROGRAMS CALLED:** CURFIT

**ERROR RETURNS:** None

**RESTRICTIONS:** A maximum of 200 points may be used to define the airfoil sections. Points must be ordered as required by subroutine CURFIT.

**SUBJECT:** FORTRAN IV Subroutine FOIL

**PURPOSE:** To transfer a normalized airfoil definition into the reference coordinate system, given the x, y, and z coordinates of the leading and trailing edges

**METHOD:** The chord length and angle of rotation, if any, are calculated. The input x/c and z/c values are multiplied by the chord length, after which they are rotated through the appropriate angle and translated to the reference coordinate system.

**USAGE:** CALL FOIL (XLE, YLE, ZLE, XTE, YTE, ZTE, XC,  
ZC, N, X, Y, Z, LE)  
DIMENSION XC(n), ZC(n), X(n), Y(n), Z(n) (N ≤ n ≤ 200)

Input:    XLE } = coordinates of the leading edge in the  
          YLE } reference coordinate system  
          ZLE }

          XTE } = coordinates of the trailing edge in the  
          YTE } reference coordinate system  
          ZTE }

          XC } = arrays of x/c and z/c values defining  
          ZC } the airfoil

          N = number of defining points

Output:    X } = arrays of x, y, and z coordinates of  
          Y } the airfoil in the reference coordinate  
          Z } system

          LE = subscript of leading edge

**SUBPROGRAMS CALLED:** NORM3  
                          ROTRAN

**ERROR RETURNS:** None

**RESTRICTIONS:** N, the number of defining points, is limited to 200.

**SUBJECT:**

**FORTRAN IV Subroutine GRAPH**

**PURPOSE:**

**To graphically display the geometric surfaces**

Since the plotting capabilities of each computer installation vary, subroutine GRAPH written at Boeing could not be used elsewhere. Therefore, subroutine GRAPH is a dummy subroutine. The information to be plotted, however, is stored on a data file where it may easily be retrieved by a plot routine tailored to the user's installation.

**SUBJECT:** FORTRAN IV Subroutine INLET

**PURPOSE:** To produce x, y, and z coordinates of the area around a fan-in-wing

**METHOD:** The area surrounding the inlet is divided into five distinct regions. The first of these, region 1 (shown in Figure 50), is the area on the upper or lower wing surface that is unaffected by the presence of the inlet.

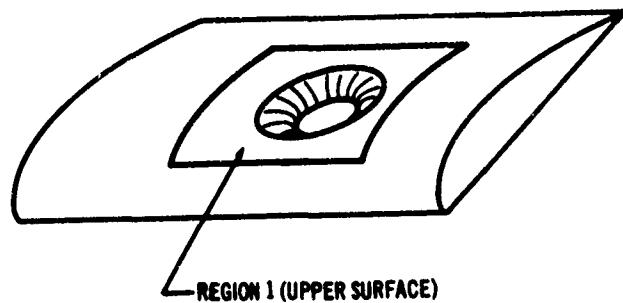


Figure 50. Subroutine INLET—Region 1 of the Inlet.

The area outside region 1 is paneled in the usual manner. Subroutine WING is usually used for this purpose. The inlet is divided into two regions, 2 and 3. Figure 51 is a side view of the inlet showing these two regions.

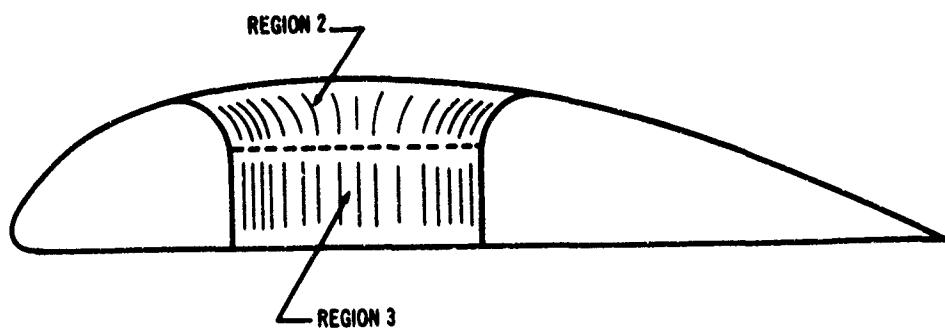


Figure 51. Subroutine INLET—Regions 2 and 3 of the Inlet.

Region 2 is usually referred to as the lip of the inlet; region 3 is the inlet throat. The two regions differ in that the throat is simply a cylinder, whereas the lip flares out to join the wing surface.

Region 4 is a disk-shaped area on the lower surface of the wing connecting region 3 (the inlet throat) to region 1 of the lower surface. Figure 52 shows a bottom view of the inlet area, showing regions 4 and 1.

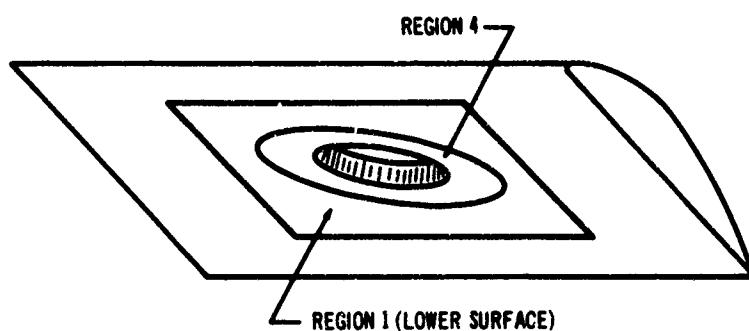


Figure 52. Subroutine INLET —Region 4 of the Inlet.

Region 5 is the internal vortex network, corresponding to the lifting system produced by subroutine LIFT. Figure 53 shows a cross section showing region 5.

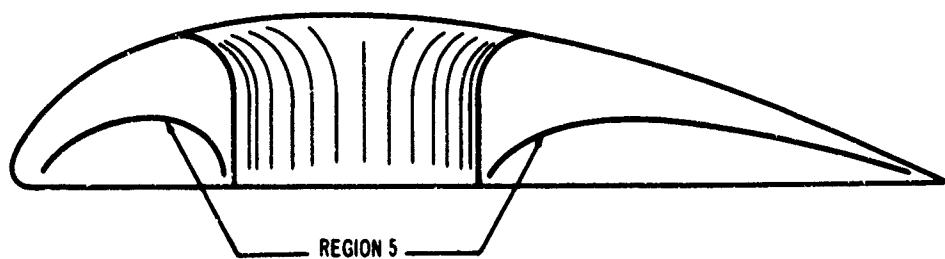


Figure 53. Subroutine INLET —Region 5 of the Inlet.

Paneling methods for each of these five regions are described below.

Before panel coordinates for any regions can be calculated, a wing definition must be input. This definition is the same as that used in the WING subroutine; namely, two tables of  $x/c$  and  $z/c$  values, and the  $x$ ,  $y$ , and  $z$  coordinates of the leading and trailing edges of the airfoil sections.

When paneling region 1, a table of percentages and two tables of  $x$  versus  $y$  are input. The  $x$ - $y$  tables are endpoint coordinates of the rays extending away from the fan. The first table gives the points most distant from the fan, while the second table gives the points nearest the fan, adjacent to region 2. The table of percentages gives the distance along the rays to define a radial panel edge. A simple region 1 is illustrated in Figure 54.

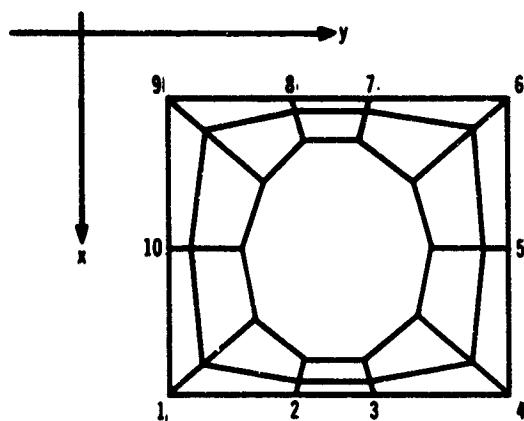


Figure 54. Subroutine INLET—Paneling Region 1.

Input for this region would be two tables of ten  $x$  and  $y$  values each, ordered in the indicated manner. The table of percentages would contain three values: 0, 0.33, and 1.00.

To panel the lip of the inlet (region 2) a table of angles and two tables of radius  $r$  versus  $z$  are input. The  $r$ - $z$  tables define two sections through the inlet. A typical section is shown in Figure 55.

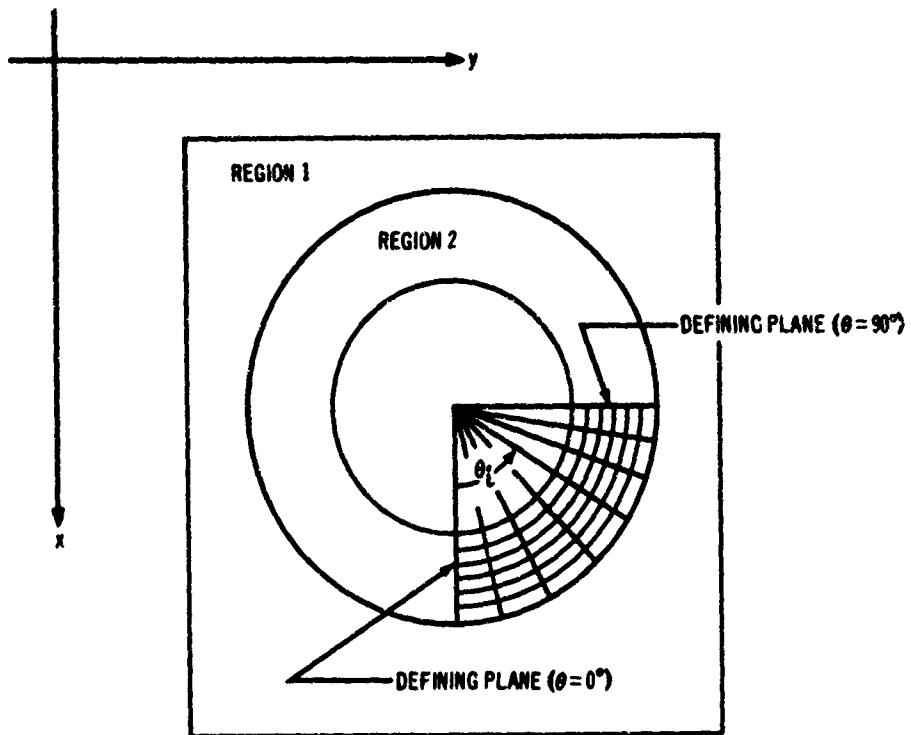


Figure 55. Subroutine INLET-- Paneling Region 2.

Each of the sections is defined on a given radial plane through the center of the inlet. Linear interpolation between these two sections is used to calculate coordinates of sections on radial planes specified by the input angles.

To panel region 3, a table of angles and two tables of z values are input. These values are z coordinates of the radial panel edges extending from the juncture between regions 2 and 3 down the throat to the lower surface of the wing. Like region 2, the desired coordinates are interpolated from the two defining tables and transformed to the reference coordinate system.

To panel region 4, two tables of r and one table of angles are input, and coordinates on the bottom surface of the wing are calculated at each angle with a radius obtained by interpolation between the two input tables of r.

Input for region 3 is identical to that for region 2. Two tables of r versus z and one table of angles are required.

In addition to the regions described above, subroutine INLET has a feature that is quite useful when inputs are prepared. After the wing definition has been input, subroutine INLET will produce x, y, and z coordinates in the reference coordinate system and r and z coordinates in the fan coordinate system of any section desired through the wing. These coordinates may be plotted, and the tables of x, y, and r that are required for regions 1 through 5 can be obtained from the resulting graphs.

**USAGE:**

CALL INLET

Input for subroutine INLET is described in detail in Section 3.1.2.

Output consists of x, y, and z coordinates of the regions requested, optionally punched on cards.

**SUBPROGRAMS  
CALLED:**

FITTER  
FOIL  
NORM2  
NORM3  
OBLIQ  
RCOSIN

**ERROR RETURNS:** None

**RESTRICTIONS:** None

**SUBJECT:**

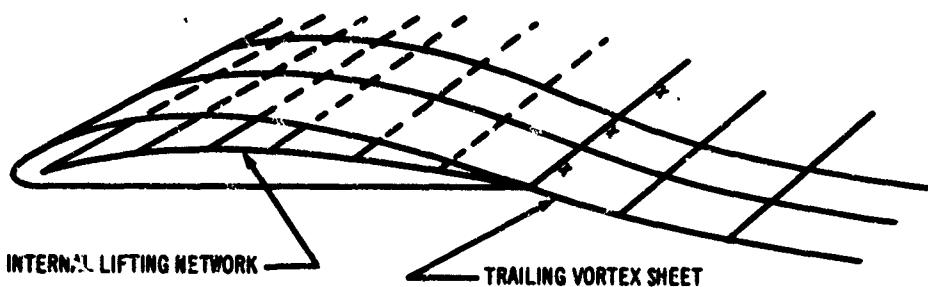
**FORTRAN IV Subroutine LIFT**

**PURPOSE:**

To produce x, y, and z coordinates of a lifting system network

**METHOD:**

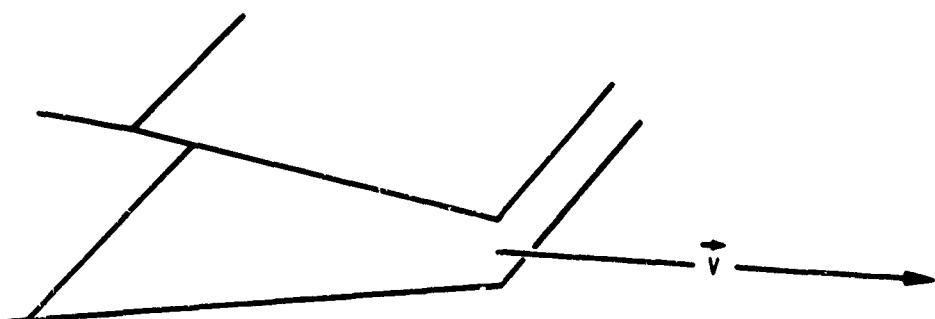
The lifting network is divided into two parts: the internal lifting system and the trailing vortex sheet, as shown in Figure 56.



**Figure 56. Subroutine LIFT—Lifting System Geometry.**

The internal system is defined by an inboard and an outboard profile and stations at which panel edges are desired. The trailing vortex sheet is defined by the length and direction cosines of each segment extending from the trailing edge of the internal network.

The x's in the figure above represent boundary points. Their locations are calculated with respect to the trailing edge of the wing, as shown in Figure 57.



**Figure 57. Subroutine LIFT—Flow at the Trailing Edge.**

Vector  $\vec{V}_1$  bisects the angle formed by the upper and lower surfaces of the wing trailing edge. The boundary point will lie on  $\vec{V}_1$  at the distance back from the trailing edge specified in the input.

The spanwise locations (y coordinates) of the boundary points are calculated by one of two methods, depending on the input. If y coordinates of panel edges are input, the subroutine interpolates between them, setting the boundary point at their average value. The other option available is to input the y coordinates of the boundary points themselves.

**USAGE:**

CALL LIFT

Input for subroutine LIFT is described in Section 3.1.2.

Output consists of x, y, and z coordinates of the entire lifting system and boundary points, optionally punched on cards.

**SUBPROGRAMS  
CALLED:**

ACS  
FOIL  
NORM2  
NORM3

**ERROR RETURNS:** None

**RESTRICTIONS:** None

**SUBJECT:** FORTRAN IV Subroutine LINER

**PURPOSE:** Given the equation of a straight line and a value of the independent variable, to compute the value of the dependent variable

**METHOD:** Let  $e_1x + e_2y + e_3 = 0$  represent the straight line, and let  $x$  be the independent variable. Subroutine LINER expects  $e_2 = -1.0$  resulting in the equations

$$y = e_1x + e_3 \quad (23)$$

and  $\frac{dy}{dx} = e_1 \quad (24)$

**USAGE:** CALL LINER (E, X, Y, J)  
**DIMENSION** E(3), Y(4)

**Input:** E(1) = coefficient of x  
 E(2) = -1.0  
 E(3) = constant term  
 X = value of independent variable x

**Output:** Y(1) = function value of x  
 Y(2) = E(1), if E(2) ≠ 0; 1.0, if E(2) = 0  
 Y(3) = 1.0, if E(2) ≠ 0; 0.0, if E(2) = 0  
 Y(4) = 0.0

J = 0, success  
 = 1, if vertical line and x = E(3)  
 = -1, if vertical line and x ≠ E(3)

**SUBPROGRAMS CALLED:** None

**ERROR RETURNS:** J, see USAGE

**RESTRICTIONS:** E(2) must be -1.0

**SUBJECT:** FORTRAN IV Program MAIN

**PURPOSE:** To read control cards and to call the appropriate subroutine

**METHOD:** Card columns 1 through 10 are read by the program as a control word. This word is compared to the index of allowable words by function DIRECT. A correct control card will cause loading and execution of the desired routines. An incorrect control card causes a comment to be printed. In either case, the program will continue to read words from the input file until either an EXIT control card or an end-of-file mark is read.

**USAGE:** Program MAIN is never referenced in a CALL statement.

**SUBPROGRAMS  
CALLED:**

AXISYM  
DATE  
DIRECT  
GRAPH  
INLET  
LIFT  
TUBE  
WING

**ERROR RETURNS:** None

**RESTRICTIONS:** None

**SUBJECT:** FORTRAN IV Subroutine NORM2

**PURPOSE:** To find the length and to normalize the components of a two-dimensional vector

**METHOD:** The length of the vector is found as the square root of the sum of the squares of the two components. The components are then normalized by dividing them by the length.

**USAGE:** CALL NORM2 (X, Y, X2, Y2, D)

Inputs:  $\begin{cases} X \\ Y \end{cases}$  = x and y components of the vector

$\begin{cases} X2 \\ Y2 \end{cases}$  = normalized x and y components of the vector

D = length of the vector

**SUBPROGRAMS  
CALLED:** None

**ERROR RETURNS:** None

**RESTRICTIONS:** None

**SUBJECT:** FORTRAN IV Subroutine NORM3

**PURPOSE:** To find the length and to normalize the components of a three-dimensional vector

**METHOD:** The vector length is found as the square root of the sum of the squares of the three components. The components are then normalized by dividing them by the length.

**USAGE:** CALL NORM3 (X, Y, Z, X2, Y2, Z2, D)

Inputs:  $\begin{matrix} X \\ Y \\ Z \end{matrix}$  = x, y, and z components of the vector

Outputs:  $\begin{matrix} X2 \\ Y2 \\ Z2 \end{matrix}$  = normalized x, y, and z components of the vector

D = length of the vector

**SUBPROGRAMS CALLED:** None

**ERROR RETURNS:** None

**RESTRICTIONS:** None

**SUBJECT:**

**FORTRAN IV Subroutine OBLIQ**

**PURPOSE:**

To find the coordinates of the intersections of a line (LM) drawn lengthwise on a cylinder that penetrates the upper and lower surfaces of a wing

The coordinates of points A and B shown in Figure 58 are found by subroutine OBLIQ.

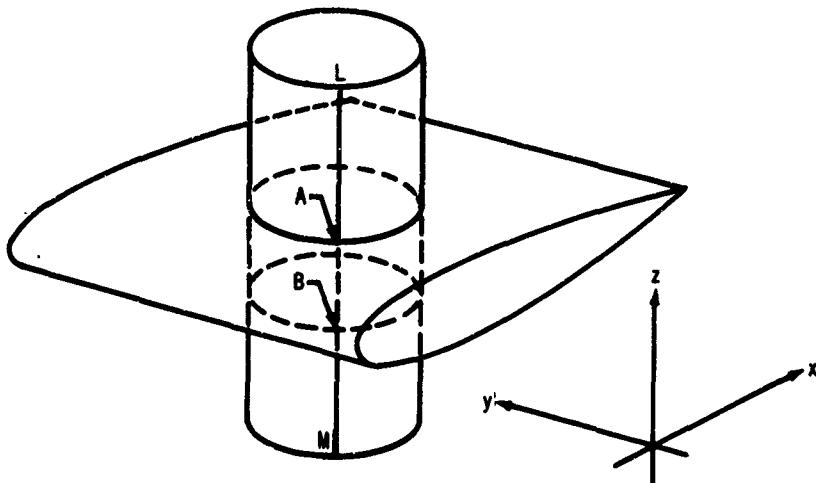


Figure 58. Subroutine OBLIQ—Wing-Cylinder Geometry.

**METHOD:**

Refer to Figure 59. The cylinder that penetrates the wing is defined by the direction cosines of its axis and the coordinates of a point on the axis. The point K is found from the length of the radius vector and its direction cosines, after which point K+1 is calculated.

The distance d from point C to point K+1 is calculated and compared to a given tolerance. If d is greater than the tolerance, point K+1 becomes point K, and new points C and K+1 are calculated. When d becomes strictly less than the given tolerance, the point on the lower surface is calculated using the same method, after which control is returned to the calling program.

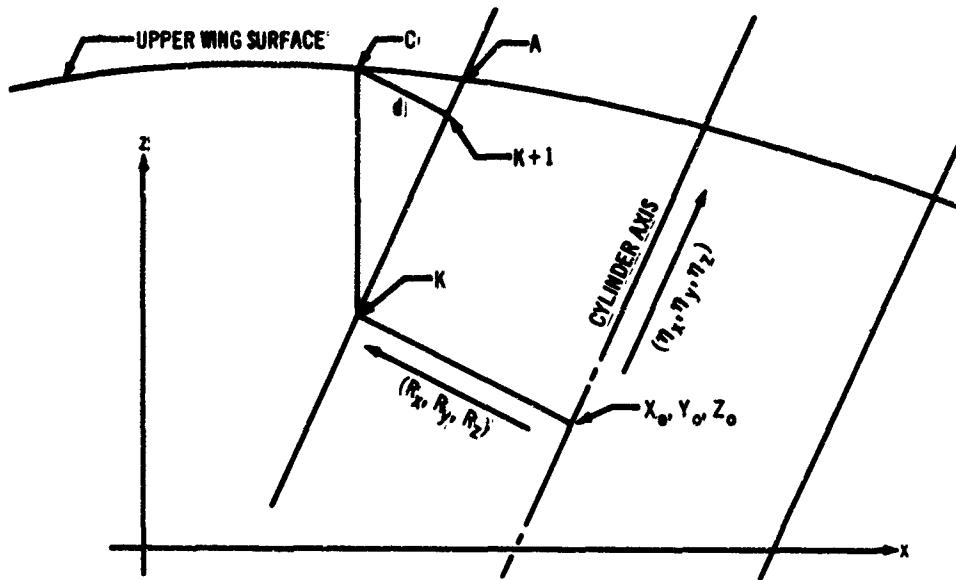


Figure 59. Subroutine OBLIQ—Wing-Cylinder Intersection.

**USAGE:**

```
CALL OBLIQ (XFOIL, YFOIL, ZFOIL, LE, N, XYZ0,
            ETAXYZ, RXYZ, R, EPS, XYZTIP)
```

```
DIMENSION XFOIL (200,2), YFOIL (200, 2), ZFOIL
(200,2), XYZ0 (3), ETAXYZ (3), RXYZ (3),
XYZTIP (3, 2)
```

**Inputs:**  $\left. \begin{matrix} XFOIL \\ YFOIL \\ ZFOIL \end{matrix} \right\}$  = tables of x, y, and z coordinates  
defining two airfoil sections

LE = subscript of leading edge

N = subscript of trailing edge (number of defining points)

XYZ0 = x, y, and z coordinates of a point on the axis of the cylinder

ETAXYZ = direction cosines of the axis of the cylinder

RXYZ = direction cosines of the radius vector

R = length of the radius vector

EPS = tolerance against which to test d

**Outputs:** XYZTIP(1, 1) }  
XYZTIP(2, 1) } = x, y, and z coordinates on  
XYZTIP(3, 1) } upper surface

XYZTIP(1, 2) }  
XYZTIP(2, 2) } = x, y, and z coordinates on  
XYZTIP(3, 2) } lower surface

**SUBPROGRAMS  
CALLED:** CURFIT

**ERROR RETURNS:** If the iterative technique described in METHOD does not converge in 50 iterations, the calling arguments are listed and the appropriate XYZTIP (upper or lower surface) is set to the last values calculated. Control is returned to the calling program directly if the failure occurred on the lower surface; otherwise, OBLIQ attempts to find the lower surface intersection before relinquishing control.

**RESTRICTIONS:** Variables must be dimensioned as shown under USAGE.

**SUBJECT:** FORTRAN IV Function PINT

**PURPOSE:** To find the intersections within a specified interval of two curves, each consisting of one or more straight-line segments

**METHOD:** Each of the curves is defined by an ordered set of points that are the end points of the line segments composing the curve.

Let S and T represent two curves of m and n segments. All of the m · n segment pairs are examined, and coordinates of intersections within the specified interval are calculated for output from PINT.

**USAGE:**

```
K = PINT (S, T, A, B, X)
DIMENSION S(2m+1), T(2n+1), X(2k)
Input:  S = array for first curve:
        m, x1, y1, ..., xm, ym
        T = array for second curve:
        n, x1, y1, ..., xn, yn
        A } = { a, b }, the specified interval of interest
        B
Output: X = array for intersections:
        x1, y1, x2, y2, ..., xk, yk
        K = number of intersections, k
```

**SUBPROGRAMS CALLED:** None

**ERROR RETURNS:** None

**RESTRICTIONS:** X values for both curves must be strictly monotonically increasing, and a < b.

SUBJECT:

FORTRAN IV Function POPO

PURPOSE:

To find the intersections of two curves, one being a polygonal arc, the other being polynomial of degree three or less

METHOD:

The polygonal arc is defined by a set of m points, which are the end points of the line segments. The polynomial is defined by its coefficients as given in the form

$$f(x) = c_1 + c_2x + c_3x^2 + c_4x^3 \quad (25)$$

Let

$$g_i(x) = a_i x + b_i y + c_i \quad (i = 1, 2, \dots, m-1) \quad (26)$$

represent the  $i^{\text{th}}$  line segment, and let

$$h_i(x) = f(x) - g_i(x) \quad (i \text{ as above}). \quad (27)$$

Subroutine ROOTP3 is called to find the real roots, if any, of  $h_i(x)$ . Those roots within the interval of interest are improved by subroutine CROSS and stored as output to the calling program.

USAGE:

N = POPO (C, S, A, B, X)

DIMENSION C(4), S(2m+1), X(2n)

Input: C = array of polynomial coefficients

S = array for polygonal arc:

m,  $x_1, y_1, \dots, x_m, y_m$

A } = the interval of interest {a, b}  
B }

Output: X = array of intersection coordinates,  
 $x_1, y_1, x_2, \dots, x_n, y_n$

N  $\geq 0$ , number of intersections, n  
= i,  $-1 \geq i \geq -7$ , error return of i from  
ROOTP3

= -8, error return of 1 from CROSS

= i,  $-9 \geq i \geq -11$ , error return of i+8  
from CROSS

SUBPROGRAMS  
CALLED:

CROSS  
CUBIT2  
LINER  
PINT  
ROOTP3

**ERROR RETURNS:** N, see USAGE

**RESTRICTIONS:** X values for the polygonal arc must be strictly monotonically increasing, and  $a < b$ .

**SUBJECT:**

**FORTRAN IV Subroutine RCOSIN**

**PURPOSE:**

To calculate the direction cosines of a radius of the lift fan, given the orientation of the fan coordinate system and the angle  $\theta$  (see Figure 60)

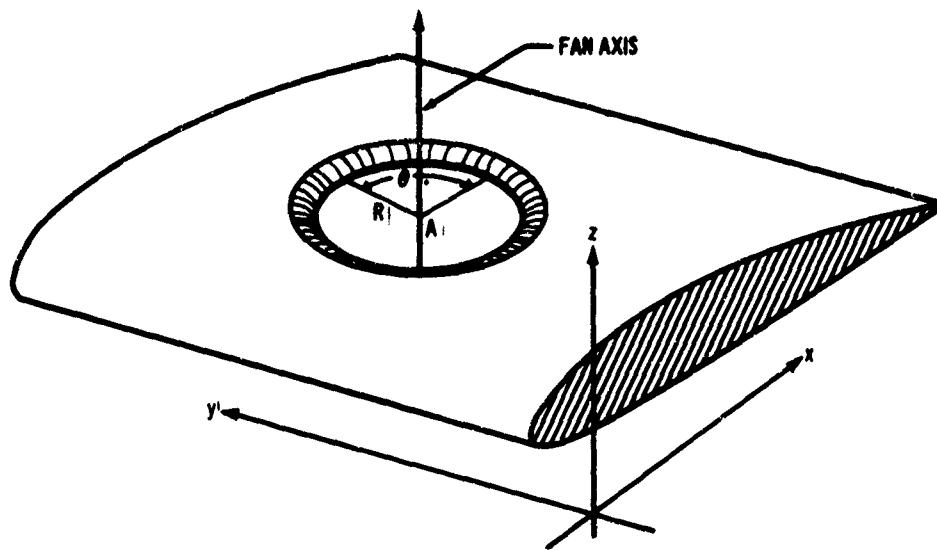


Figure 60. Subroutine RCOSIN— Lift-Fan Orientation.

**METHOD:**

The fan coordinate system is related to the reference coordinate system by the transformation matrix:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}_{\text{reference}} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}_{\text{fan}} \quad (28)$$

The angle  $\theta$  is measured in the fan plane as shown above. The reference point for  $\theta = 0$  is the line formed by the intersection between the fan plane and a plane passing through point A, parallel to the x-z plane. This line begins at point A and extends to the rear of the fan.

The direction cosines of R, the radius vector, are calculated from the formulas below:

$$\begin{bmatrix} R_x \\ R_y \\ R_z \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix} \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} \quad (29)$$

where  $R_x$ ,  $R_y$ , and  $R_z$  are the direction cosines.

**USAGE:**

CALL RCOSIN (ALPHA, THETA, RXYZ)

DIMENSION ALPHA (3,3), RXYZ (3)

Input: ALPHA = the transformation matrix  
THETA =  $\theta$ , as described in METHOD

Output: RXYZ = direction cosines of the radius vector

**SUBPROGRAMS**

**CALLED:** None

**ERROR RETURNS:** None

**RESTRICTIONS:** Theta must be input in radians.

**SUBJECT:** FORTRAN IV Subroutine ROOTP2

**PURPOSE:** To find the roots of  $ax^2 + bx + c = 0$ , any second-degree polynomial with real coefficients

**METHOD:** After checking for the cases when  $a = 0$  and when  $b^2 - 4ac < 0$ , the subroutine finds a first approximation to the root of larger magnitude,  $x_1$ , by

$$x_1 = \frac{-b \operatorname{sign}(b) \sqrt{b^2 - 4ac}}{2a} \quad (30)$$

and then obtains the root of smaller magnitude,  $x_2$ , by

$$x_2 = \frac{c}{ax_1} \quad (31)$$

A better value for  $x_1$  can generally be found by using

$$x = -x_2 - \frac{b}{a} \quad (32)$$

and subroutine ROOTP2 alternately uses Equations (31) and (32) until two successive values of  $x$  are equal.

**USAGE:** CALL ROOTP2 (C, X, I)

DIMENSION C(3), X(2,2)

Input:  $\begin{cases} C(1) \\ C(2) \\ C(3) \end{cases}$  = the coefficients of the quadratic equation  
 $C(1)x^2 + C(2)x + C(3) = 0$

Output:  $X(1,1)$  = real part of larger root  
 $X(1,2)$  = imaginary part of larger root  
 $X(2,1)$  = real part of smaller root  
 $X(2,2)$  = imaginary part of smaller root

I = -1,  $C(1) = C(2) = 0$   
= 0, imaginary parts of roots are zero  
= 1, imaginary parts of roots are not zero  
= 2, iteration to find real roots does not converge

**SUBPROGRAMS**

**CALLED:** None

**ERROR RETURNS:** I, see USAGE

**RESTRICTIONS:** None

**SUBJECT:** FORTRAN IV Function ROOTP3

**PURPOSE:** To find those real roots of a cubic within a specified interval, if any

**METHOD:** By examining the function value of the cubic at the end points of the interval and at any extrema that lie within the interval, function ROOTP3 either encounters a root or determines whether any real roots are in the interval. Generally, an iterative scheme is employed to find a real root in the interval. After one root,  $x_1$ , has been found, the quadratic resulting from dividing the cubic by  $(x-x_1)$  is solved by subroutine ROOTP2. The roots of the resultant quadratic, if real and in the interval, are substituted into the cubic and improved by subroutine CROSS.

**USAGE:** N = ROOTP3 (A, B, C, X)  
**DIMENSION** C(4), X(3)

**Input:** A { B } = the interval of interest [a, b]  
C(1) {  
C(2) {  
C(3) {  
C(4) { coefficients of the cubic equation  
 $y = C(1) + C(2)x + C(3)x^2 + C(4)x^3$

**Output:** X = array of x values of the real roots in the interval

N = the number of real roots in the interval, n  
= -1, ROOTP2 returned -1 searching for cubic extrema  
= -2, ROOTP2 returned 2 searching for cubic extrema  
= -3, ROOTP2 returned -1 solving for resulting quadratic  
= -4, ROOTP2 returned 2 solving for resulting quadratic  
= -5, ROOTP2 returned -1 solving quadratic when C(4) = 0  
= -6, ROOTP2 returned 2 solving quadratic when C(4) = 0  
= -7, CROSS unable to find real root known to be in the interval

**SUBPROGRAMS  
CALLED:**

CROSS  
CUBIT2  
LINER  
ROOTP2

**ERROR RETURNS:** N, see USAGE

**RESTRICTIONS:**  $a \leq b$

**SUBJECT:** FORTRAN IV Subroutine ROTRAN

**PURPOSE:** To rotate and translate a given set of x and y coordinates from one coordinate system to another

**METHOD:** The given input coordinates are rotated about the origin of their coordinate system a given number of radians. The resulting coordinates are then translated.

**USAGE:**

```
CALL ROTRAN (XIN, YIN, N, SIN, COS, X0, Y0, XOUT,  
             YOUT)
```

DIMENSION XIN(n), YIN(n), XOUT(n), YOUT(n) (N ≤ n)

Input: XIN } = x and y coordinates to be  
 YIN } transformed

N = the number of points, n

SIN } = the sine and cosine of the rotation angle  
COS }

X0 } = coordinates of the origin of the first  
Y0 } system with respect to the second

Output: XOUT } = x and y coordinates in the second  
 YOUT } system

**SUBPROGRAMS CALLED:** None

**ERROR RETURNS:** None

**RESTRICTIONS:** None

**SUBJECT:** FORTRAN IV Subroutine SCONIC

**PURPOSE:** To find the arc length,  $s$ , of a conic over a specified interval  $[a, b]$

Since conic efflux tube trajectories are not considered, subroutine SCONIC is not used. It is included as a dummy subroutine.

**SUBJECT:** FORTRAN IV Subroutine SCUBIC

**PURPOSE:** To find the arc length,  $s$ , of a cubic over a specified interval  $[a, b]$

**METHOD:** Let  $y = c_1 + c_2x + c_3x^2 + c_4x^3$  be the equation of the cubic. The desired arc length is

$$s = \int_a^b \left[ 1 + \left( \frac{dy}{dx} \right)^2 \right]^{1/2} dx \quad (33)$$

Subroutine SCUBIC evaluates this integral numerically using subroutine SIMPRC.

**USAGE:** CALL SCUBIC (C, A, B, EPS, S, K)

DIMENSION C(4)

Input:  $\begin{cases} C(1) \\ C(2) \\ C(3) \\ C(4) \end{cases}$  coefficients of the cubic,  $c_1, c_2, c_3,$   
and  $c_4$

$\begin{cases} A \\ B \end{cases}$  the interval  $[a, b]$  for which the arc  
length is desired

tolerance required by SIMPRC to  
EPS = terminate iteration for evaluating the  
integral

Output: S = the arc length,  $s$

K = 1, success  
= 2, SIMPRC failed to converge while  
evaluating the integral

**SUBPROGRAMS**

**CALLED:**

SIMPRC  
VALCUB

**ERROR RETURNS:** K, see USAGE

**RESTRICTIONS:** None

SUBJECT:

#### FORTRAN IV Subroutine SELECT

PURPOSE:

To compute the z coordinates of the line segment used by function POPO

METHOD:

Subroutine SELECT calculates a maximum value of z or  $z_{\max}$ , and a minimum value of z or  $z_{\min}$ , one of two ways, depending on the amount of rotation the coordinate system is to undergo. For rotations other than in the interval  $[225^\circ, 315^\circ]$ , the method illustrated in Figure 61 is used.

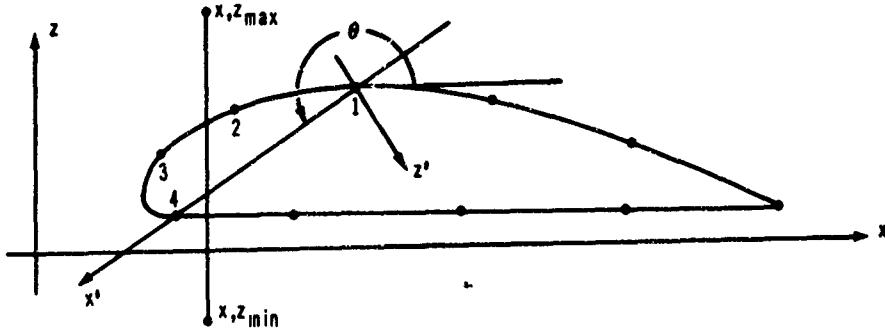


Figure 61. Subroutine SELECT—Airfoil Coordinate Calculation.

Let points 1 through 4 be the four points used as input for BQUAD2. Subroutine SELECT chooses  $z_{\max}$  to be the maximum z coordinate of the four points plus the distance between points 1 and 4, while  $z_{\min}$  is the minimum z minus the distance.

For angles of rotation in the interval  $[225^\circ, 315^\circ]$ ,  $z_{\max}$  is calculated at the z coordinate of point 2 plus  $1/15$  the distance between points 1 and 4, while z is the z coordinate of point 3 minus  $1/15$  the minimum distance.

USAGE:

CALL SELECT (ZC, ZMX, ZMN, D, SIN, K)

DIMENSION ZC(n) ( $n \geq K$ )

Input: ZC = a vector of n values for z

D = distance between points 1 and 4, above

SIN = sine of the angle of rotation,  $\theta$

K = subscript in ZC vector of z value  
immediately prior to the interval of interest

Output:  $ZMX = \begin{cases} z_{\max} \\ z_{\min} \end{cases}$ , as described in METHOD

**SUBPROGRAMS**  
**CALLED:** None  
**ERROR RETURNS:** None  
**RESTRICTIONS:** None

**SUBJECT:** FORTRAN IV Subroutine SIMPRC  
**PURPOSE:** To evaluate the integral of any function,  $y = f(x)$ , over the interval  $[a, b]$  using Simpson's Rule  
**METHOD:** Let the integral be  $F = \int_a^b f(x) dx$ .  
 The subroutine evaluates the integral with the formula

$$F = \int_a^b f(x) dx = \frac{h}{3} (y_0 + 4y_1 + 2y_2 + 4y_3 + \dots + 2y_{2n-2} + 4y_{2n-1} + y_{2n}) \quad (34)$$

where  $2n = \text{number of intervals}$   
 $h = (b-a)/2n$   
 $y_i = f(a + ih)$

The subroutine first finds an  $F_1$  and an  $F_2$  setting,  $2n = 4$  and  $8$ , respectively. These two values for the integral are then compared with a given tolerance,  $\epsilon$ , according to Equation (35) or (36).

$$\frac{|F_{i-1} - F_i|}{\max(|F_i|, |F_{i-1}|)} < \epsilon \quad (35)$$

or

$$|F_i - F_{i-1}| < |\epsilon| \quad (36)$$

Equation (35) is used when  $\epsilon \leq 0$ ; otherwise, Equation (36) is used.

If neither formula is satisfied, the number of intervals is doubled and a new  $F_i$  is found. When the convergence test is satisfied, the last  $F_i$  calculated,  $F_n$ , is adjusted by the formula

$$F = F_n + 1/15 (F_n - F_{n-1}) \quad (37)$$

The function to be integrated,  $f(x)$ , is computed by subroutine VALCUB, and is given below:

$$f(x) = \left[ 1 + \left( c_2 + 2c_3x + 3c_4x^2 \right)^2 \right]^{1/2} \quad (38)$$

where  $c_2$ ,  $c_3$ , and  $c_4$  are the cubic coefficients in the equation  $y = c_1 + c_2x + c_3x^2 + c_4x^3$ . This cubic defines the trajectory of the jet efflux tube.

**USAGE:** CALL SIMPRC (A, B, C, VALCUB, E, F, J)  
DIMENSION C(4)

**Inputs:** A } = the limits of integration {a, b}  
B }  
  
C(1) }  
C(2) } = the coefficients  $c_1$  through  $c_4$  of the  
C(3) } cubic defining the efflux tube trajectory  
C(4) }

VALCUB = the name of the subroutine that evaluates the integrand

E = the desired tolerance,  $\epsilon$

**Outputs:** F = the value of the integral

J = 1, success  
= 2, iteration does not converge  
= -2, error return from VALCUB = -J

**SUBPROGRAMS  
CALLED:** VALCUB

**ERROR RETURNS:** J, see USAGE

**RESTRICTIONS:** The integrand must be finite within the interval [a, b].

**NOTE:** An EXTERNAL card containing the name VALCUB must appear in the calling routine.

**SUBJECT:** FORTRAN IV Function SQUAD

**PURPOSE:** To compute the arc length,  $s$ , of a quadratic

Because quadratic efflux tube trajectories are not considered, function SQUAD is not used. It is included as a dummy subroutine.

**SUBJECT:** FORTRAN IV Function TRACON  
**PURPOSE:** To transform coefficients of an explicit conic to those of  
the corresponding implicit conic, or vice versa

Because conic efflux tube trajectories are not considered,  
function TRACON is not used. It is included as a dummy  
subroutine.

**SUBJECT:**

**FORTRAN IV Subroutine TRNSLT**

**PURPOSE:**

To translate and rotate a set of x and y coordinates from a given coordinate system to a second coordinate system

**METHOD:**

After translating the given points to the second coordinate system, the subroutine rotates them through a given angle.

**USAGE:**

CALL TRNSLT (XIN, YIN, N, X0, Y0, SIN, COS, XOUT, YOUT)

DIMENSION XIN(n), YIN(n), XOUT(n), YOUT(n), (N ≤ n)

**Input:**       $\begin{cases} XIN \\ YIN \end{cases}$  = x and y coordinates to be transformed

N = the number of points, n

$\begin{cases} X0 \\ Y0 \end{cases}$  = coordinates of the origin of the first system with respect to the second

$\begin{cases} SIN \\ COS \end{cases}$  = the sine and cosine of the angle of rotation

**Output:**  $\begin{cases} XOUT \\ YOUT \end{cases}$  = x and y coordinates of the second system

**SUBPROGRAMS**

**CALLED:**

None

**ERROR RETURNS:** None

**RESTRICTIONS:** None

SUBJECT: FORTRAN IV Subroutine TUBE  
 PURPOSE: To calculate the x, y, and z coordinates of a jet efflux tube  
 METHOD: Refer to Figure 62.

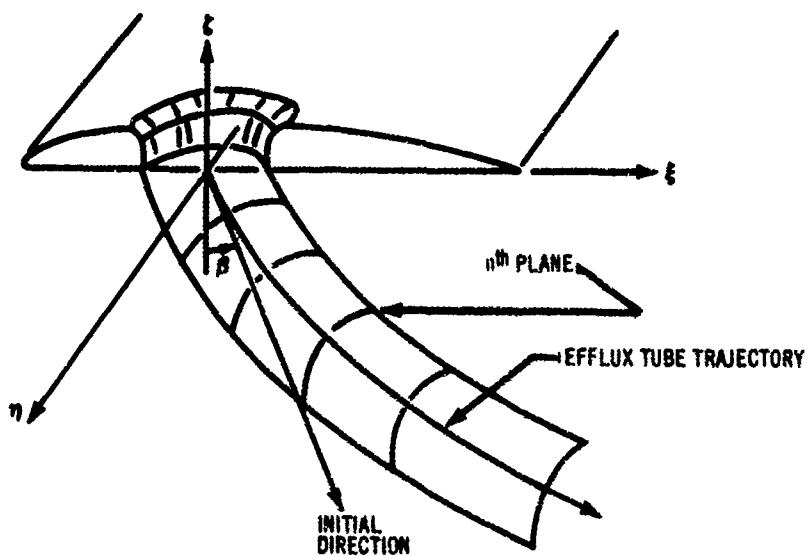


Figure 62. Subroutine TUBE—Jet Efflux Tube Geometry.

The trajectory of the efflux tube is found from the equation

$$\frac{\xi}{D} = - \frac{0.25}{\cos \beta} \left( \frac{V_\infty}{V_j} \right)^2 \left( \frac{t}{D} \right)^3 - \left( \frac{t}{D} \right) \tan \beta, \quad (39)$$

where  $D$  = exit diameter of the fan

$\beta$  = deflection angle

$V_\infty$  = free-stream velocity

$V_j$  = jet exit velocity

The origin of the  $(\xi, \eta, \xî)$  coordinate system is the point at which the fan axis intersects a plane normal to it, at the average  $z$  coordinate of the points input as tube-wing intersections.

The coordinates of the points that define the efflux tube are calculated thus. The panel edges oblique to the direction of the trajectory are defined by a series of planes cutting the efflux tube. The first plane is parallel to the lower surface of the wing, the  $n^{\text{th}}$  plane is normal to the efflux tube trajectory, and the intermediate planes vary linearly between these two. All planes downstream from the  $n^{\text{th}}$  plane are normal to the trajectory.

In general, the lower surface of the wing will not be a plane, with the result that all panel-defining "planes" upstream from the  $n^{\text{th}}$  plane are also nonplanar.

The panel edges parallel to the trajectory begin at the points input as the intersection between the wing and the tube, as shown in Figure 63.

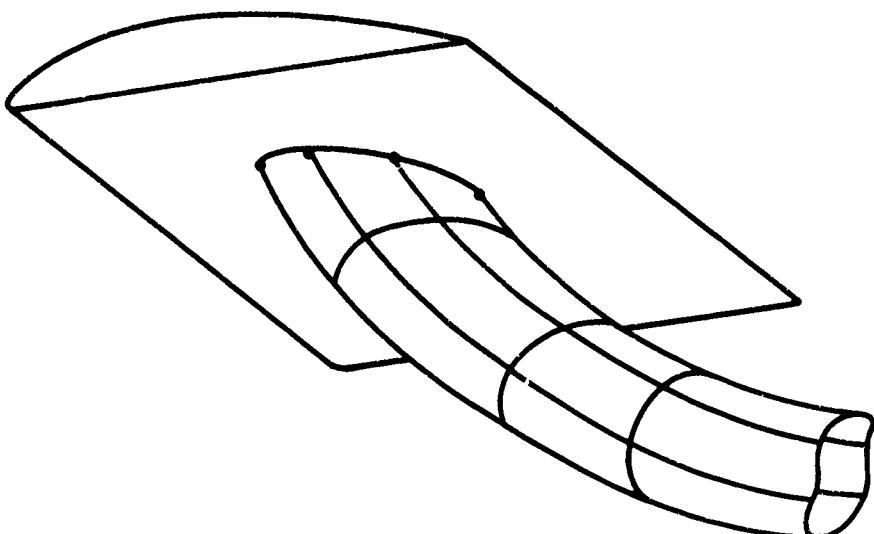


Figure 63. Subroutine TUBE—Tube Paneling.

**USAGE:**

**CALL TUBE**

Input for subroutine TUBE is described in Section 3.1.2.

Output consists of x, y, and z coordinates of the efflux tube, optionally punched on cards.

**SUBPROGRAMS**  
**CALLED:** ACS  
**ERROR RETURNS:** None  
**RESTRICTIONS:** None

**SUBJECT:** FORTRAN IV Subroutine VALCUB

**PURPOSE:** To evaluate the function

$$f(x) = [1 + (c_2 + 2c_3x + 3c_4x^2)^2]^{1/2} \quad (40)$$

for a given value of x

**METHOD:** Evaluation of the above equation

**USAGE:** CALL VALCUB (X, U, V, W, Y, K)

DIMENSION U(3)

Inputs:    U(1)  
              U(2)  
              U(3) } = coefficients  $c_2$ ,  $c_3$ , and  $c_4$

X = the value of the independent variable x

Output:    Y = the function value at x = X

K = 1

V and W are not used

**SUBPROGRAMS**

**CALLED:** None

**ERROR RETURNS:** None

**RESTRICTIONS:** None

**SUBJECT:** FORTRAN IV Subroutine WING  
**PURPOSE:** To calculate the x, y, and z coordinates of a wing  
**METHOD:** The wing is defined by two airfoil sections, inboard and outboard. Each section is input as a table of x/c versus z/c values, as shown in Figure 64.

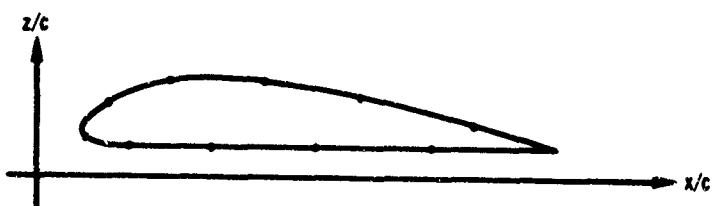


Figure 64. Subroutine WING— Defining Airfoil Section.

The position of these sections in the reference coordinate system is input, and the two section definitions are transformed to that system, as shown in Figure 65.

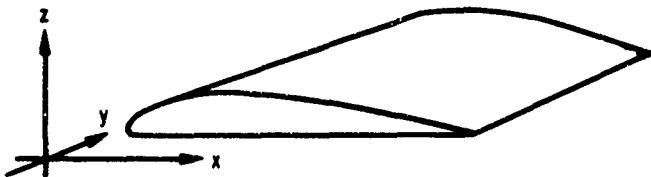


Figure 65. Subroutine WING— Wing Geometry.

The spanwise panel edge coordinates are calculated by one of two methods: either the defining x/c and z/c values are used, or new x/c values are input, with the corresponding z/c values calculated by subroutine CURFIT. The streamwise panel edge coordinates are simply input y values. The intersections of the spanwise and streamwise lines define the corners of the panels, as shown in Figure 66.

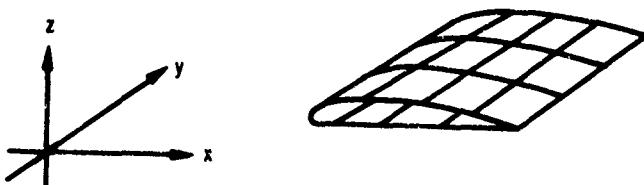


Figure 66. Subroutine WING— Wing Paneling.

**USAGE:** CALL WING  
Input for subroutine WING is described in Section 3.1.2.  
Output consists of x, y, and z coordinates of panel corners.

**SUBPROGRAMS  
CALLED:** CURFIT  
FOIL  
XCN2

**ERROR RETURNS:** None

**RESTRICTIONS:** None

**SUBJECT:** FORTRAN IV Subroutine XCN2

**PURPOSE:** To exchange x and y coordinates of a point with those of a second point

**METHOD:** The x coordinate of point 1 is stored temporarily in a dummy location. The x coordinate of point 2 is placed in the location for the x coordinate of point 1, and the dummy (x coordinate of point 1) is placed in the location for the x coordinate of point 2.  
The process is repeated for the y coordinates of the two points.

**USAGE:** CALL XCN2 (X1, Y1, X2, Y2)

Input:  $X1 \left\{ \begin{array}{l} \\ Y1 \end{array} \right\}$  = coordinates of point 1  
 $X2 \left\{ \begin{array}{l} \\ Y2 \end{array} \right\}$  = coordinates of point 2

Output:  $X1 \left\{ \begin{array}{l} \\ Y1 \end{array} \right\}$  = coordinates of point 2  
 $X2 \left\{ \begin{array}{l} \\ Y2 \end{array} \right\}$  = coordinates of point 1

**SUBPROGRAMS CALLED:** None

**ERROR RETURNS:** None

**RESTRICTIONS:** None

B. Potential-flow program subroutine descriptions. — The subroutines listed alphabetically below are discussed in this section.

**Subroutine Index**

<u>Subroutine</u>	<u>Page</u>	<u>Subroutine</u>	<u>Page</u>
AERO	154	PLOT	232
AEROSL	159	QVGEO	233
CONSUR	162	READBP	237
CONTRL	170	READTP	238
DATA	172	RHS	239
DATE	173	SBP	241
ENDSL	174	SEARCH	243
FAIL	176	SFLOW	248
FLOW	179	SFLOW1	256
FORCES	183	SNF	262
GEOM	208	SORDQ	264
INTURP	210	SOREQ	274
MAIN	211	SPGEO	281
MATDAT	216	STREAM	284
MATRIX	217	UFLOW	289
MHVGEO	221	UFLOW1	295
OFFPTS	225	VORDQ	301
PANEL	227	VOREQ	305
PARTIN	231	WRTETP	309

**SUBJECT:****FORTRAN IV Subroutine AERO****PURPOSE:**

To calculate the normal velocity matrices  $[VN_{ij}]$  and  $[B_{ik}]$  and to store them in a data file

**METHOD**

The boundary condition of parallel flow to the model's surface at each boundary point is used to establish a matrix equation for the unknown singularity strengths:

$$[VN_{ij}] \quad [\sigma_{jk}] = [B_{ik}] \quad (41)$$

where  $VN_{ij}$  = the normal component of velocity at the  $i^{\text{th}}$  boundary point due to the  $j^{\text{th}}$  unit strength singularity

$\sigma_{jk}$  = the strength of the  $j^{\text{th}}$  singularity that satisfies the  $k^{\text{th}}$  solution

$B_{ik}$  = the required normal component of velocity at the  $i^{\text{th}}$  boundary point for the  $k^{\text{th}}$  solution. It is the sum of the free-stream normal velocity component and the specified normal velocity.

The matrices  $[VN_{ij}]$  and  $[B_{ik}]$  are stored in a data file for use when the unknown singularity strengths are found. Because of the limited memory size (core) of the computer, all matrices used in this program are partitioned and stored in data files in partition form. The routines READTP and WRTETP transfer the matrix partitions between core and the data files.

Before calculating the matrices, subroutine AERO reads the desired solutions from the input data file. Each solution is either an angle of attack ( $\alpha$ )/angle of yaw ( $\psi$ ) specification or a zero free-stream velocity specification. If the  $\alpha/\psi$  angles are specified, the free-stream velocity components for the  $k^{\text{th}}$  solution are given by

$$V_{\infty x_k} = \cos \alpha_k \cos \psi_k$$

$$V_{\infty y_k} = -\cos \alpha_k \sin \psi_k \quad (42)$$

$$V_{\infty z_k} = \sin \alpha_k$$

(Note that the free-stream velocity is equal to unity.)

If the zero free-stream velocity condition is specified, the components are

$$\left. \begin{array}{l} V_{\infty x_k} \\ V_{\infty y_k} \\ V_{\infty z_k} \end{array} \right\} = 0.0 \quad (43)$$

The free-stream velocity components are used by subroutine RHS to calculate partitions of the  $[B_{ik}]$  matrix.

The matrix  $[VN_{ij}]$ , a function of the velocity component matrices and the unit normal vector components, is given by

$$[VN_{ij}] = [VX_{ij}] \left\{ n_{x_i} \right\} + [VY_{ij}] \left\{ n_{y_i} \right\} + [VZ_{ij}] \left\{ n_{z_i} \right\} \quad (44)$$

where  $VX_{ij}$   
 $VY_{ij}$   
 $VZ_{ij}$

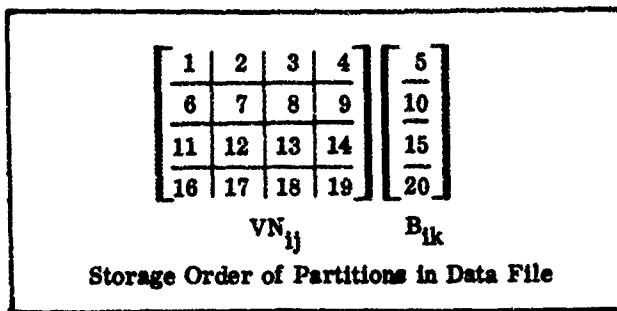
= velocity components at the  $i^{\text{th}}$  boundary point due to the  $j^{\text{th}}$  unit strength singularity

$n_{x_i}$   
 $n_{y_i}$   
 $n_{z_i}$

= unit normal vector components at the  $i^{\text{th}}$  boundary point

The velocity component matrices are computed by subroutine SFLOW if the flow is symmetrical or by subroutine UFLOW if the flow is unsymmetrical. The unit normal vector components are obtained from the boundary point quantities data file.

The partitions of the  $[VN_{ij}]$  and  $[B_{ik}]$  matrices are stored in a data file in the order of the example illustrated in Figure 67.



**Figure 67. Subroutine AERO—Normal Velocity Matrices.**

Subroutine AERO's last function is to call subroutine MATDAT, which stores the matrix operating instructions in a data file. The Boeing-developed matrix routines use these instructions to obtain the singularity strengths and the resultant-induced velocity components.

For temporary storage, subroutine AERO uses the labeled COMMON block KOM3 to store the matrix partitions. After a partition of the [VN<sub>ij</sub>] or [B<sub>ik</sub>] matrix has been calculated, it is immediately transferred from KOM3 to a data file. The unit normal vector components for a single partition are stored in the boundary-point quantities array, BP, which appears in the labeled COMMON block KOM2.

**USAGE:**

COMMON (See program MAIN for blank COMMON description.)

CALL AERO

Input: TITLE = any desired title

COM(1) } = date of run  
COM(2) }

ICOM(1) = 1, symmetric flow  
= 3, unsymmetric flow

ICOM(2) = maximum matrix partition size

ICOM(9) = matrix operating instruction data  
file indicator

NT1 = data file number of the normal  
velocity matrices [VN<sub>ij</sub>] and [B<sub>ik</sub>]

**NTSIN** = input data file number  
**NTSOUT** = output data file number  
**NT7** = data file number of the matrix operating instructions  
**NT8** = data file number of the boundary-point quantities  
**NT10** = data file number of the singularity-defining quantities  
**NT11** = data file number of the velocity component matrix [ $VX_{ij}$ ]  
**NT12** = data file number of the velocity component matrix [ $VY_{ij}$ ]  
**NT13** = data file number of the velocity component matrix [ $VZ_{ij}$ ]  
**NPR** = number of row partitions in the velocity component matrices  
**NPC** = number of column partitions in the velocity component matrices  
**Output:**  
**ICOM(3)** = force and moment option code  
**ICOM(4)** = plot option code  
**ICOM(7)** = number of lift fans in the model  
**ICOM(8)** = number of streamlines desired  
**NRHS** = number of simultaneous solutions desired  
**ALPHA** = array of angles of attack,  $\alpha_k$   
**PSI** = array of angle of yaw,  $\psi_k$   
**FSVX**  
**FSVY**  
**FSVZ** } = arrays of free-stream velocity components,  $(V_\infty x_k, V_\infty y_k, V_\infty z_k)$

**SUBPROGRAMS  
CALLED:**

SFLOW  
UFLOW  
RHS  
READTP  
WRTETP  
MATDAT

**ERROR RETURNS:** None

**RESTRICTIONS:** A maximum of five simultaneous solutions may be specified.

SUBJECT:

FORTRAN IV Subroutine AEROSL

PURPOSE:

To calculate the flow properties along a streamline

METHOD:

This routine calls subroutine SFLOW1 or UFLOW1, depending on whether the flow is symmetric or unsymmetric, to calculate, at selected points, the velocity derivative normal to the streamline. The resultant velocity components used to determine the velocity derivatives are the sum of the singularity-induced velocity components and the free-stream velocity components. The components at the  $i^{\text{th}}$  streamline point for the  $k^{\text{th}}$  solution are given by

$$\begin{aligned} V_{xbo_i} &= u_i + V_\infty x_k \\ V_{ybo_i} &= v_i + V_\infty y_k \\ V_{zbo_i} &= w_i + V_\infty z_k \end{aligned} \quad (45)$$

The velocity component normal to the streamline is given by

$$\widehat{V}_{bo_i} \cdot \widehat{b}_i = b_{x_i} V_{xbo_i} + b_{y_i} V_{ybo_i} + b_{z_i} V_{zbo_i} \quad (46)$$

where  $b_{x_i}$  }  
 $b_{y_i}$  }  
 $b_{z_i}$  } = unit normal vector components at the  
i<sup>th</sup> streamline point

Finally, the velocity derivative is determined from finite difference by

$$\left( \frac{dV_t}{ds_t} \right)_i = \frac{\widehat{V}_{bo_i} \cdot \widehat{b}_i}{\epsilon t_{\max}} \quad (47)$$

where  $\epsilon$  = the specified fraction of the longest source-panel diagonal

$t_{\max}$  = the longest source-panel diagonal

At every streamline point, the accumulative length of the streamline, the velocity magnitude at the boundary point of the source panel containing the streamline point, and the velocity derivative are punched on cards (3E10.3 format) as input to the boundary-layer program.

USAGE:

COMMON (See program MAIN for blank COMMON description.)

COMMON /SL3/SS (50), XS (50), YS (50), ZS (50), BX (50),  
BY (50), BZ (50), XBO (50), YBO (50), ZBO (50), VF (50)

CALL AEROSL (KSL, KSOL, MSL)

Input: ICOM(1) = 1, symmetric flow  
= 3, unsymmetric flow

NT1 = data file number of the singularity strengths

NTSOUT = output data file number

NT10 = data file number of the singularity-defining quantities

NT14 = punched card output data file number

SS = array of the accumulative length of the streamline at each of the streamline points

BX } BY } BZ } = arrays of the streamline unit normal vector components

XBO } YBO } ZBO } = arrays of the coordinates of the streamline points in the reference coordinate system

VF = array of the velocity magnitudes at the boundary points of the source panels containing the streamline points

KSL = streamline number

KSOL = solution number

MSL = number of streamline points

**SUBPROGRAMS**

**CALLED:** SFLOW1  
UFLOW1

**ERROR RETURNS:** None

**RESTRICTIONS:** None

**SUBJECT:** FORTRAN IV Subroutine CONSUR  
**PURPOSE:** To calculate the flow properties on the lift-fan barrier  
**METHOD:** A lift fan consists of a barrier (fan surface), centerbody, exit plane (flow exiting from the lower surface), and centerbody base, as shown in Figure 68.

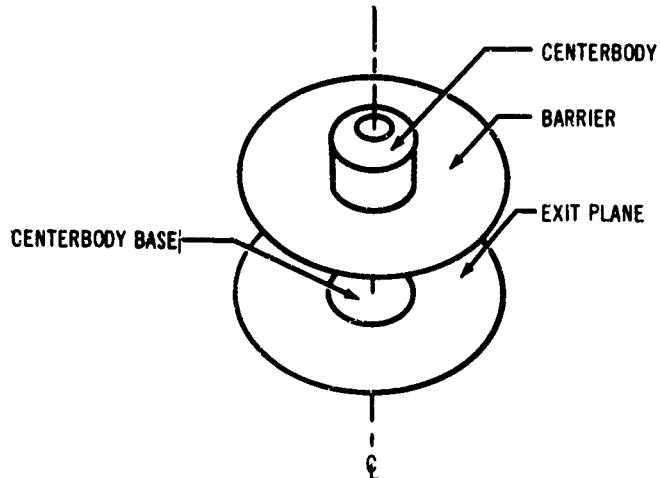


Figure 68. Subroutine CONSUR— Lift-Fan Assembly.

The potential-flow model can have a maximum of ten lift fans. Depending on its location, either half or the complete fan is defined. Only half of the fan is required when the axis of the fan is in the plane of symmetry ( $x - z$ ). For unsymmetric flow, the complete fan must be defined.

This routine calculates the velocities and pressure coefficients at the boundary points on the barrier, which is composed of quadrilateral vortex and multihorseshoe vortex singularities. Arrangement of the barrier singularities is shown in Figure 69. As viewed from above, the barrier is subdivided into  $N$  columns ( $j = 1, \dots, N$ ) with each column containing  $M$  singularities ( $i = 1, \dots, M$ ).

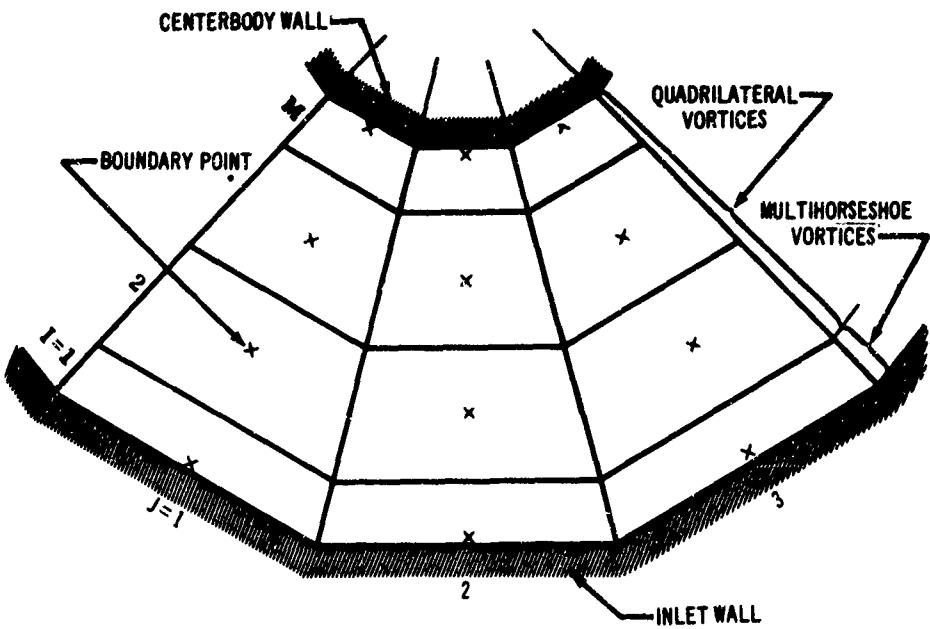


Figure 69. Subroutine CONSUR — Barrier Singularities.

The radial vortex segments of the barrier induce a velocity component in the circumferential direction.

$$v_{c_{i,j}} = \frac{\Delta \sigma_{r_{i,j}}}{l_{r_{i,j}}}, \quad i = 1, \dots, M \\ j = 1, \dots, N+1 \quad (48)$$

where  $\Delta \sigma_{r_{i,j}} = \sigma_{i,j} - \sigma_{i,j-1}$ , the net singularity strengths

$$l_{r_{i,j}} = \sqrt{(x_{i,j} - x_{i,j-1})^2 + (y_{i,j} - y_{i,j-1})^2 + (z_{i,j} - z_{i,j-1})^2},$$

the distance between boundary points

This velocity is computed at the midpoint of the radial line segment. By averaging the velocities at the four neighboring points, the value at the midpoint of the circumferential line segments is found.

$$v_{ca_{i,j}} = \frac{1}{4} [v_{c_{i+1,j}} + v_{c_{i,j}} + v_{c_{i+1,j+1}} + v_{c_{i,j+1}}], \quad (49)$$

$i = 1, \dots, M-1$   
 $j = 1, \dots, N$

The circumferential vortex segments induce a velocity component in the radial direction.

$$v_{r_{i,j}} = \frac{\Delta \sigma_{c_{i,j}}}{l_{c_{i,j}}}, \quad i = 1, \dots, M-1 \quad (50)$$

$i = 1, \dots, N$

where  $\Delta \sigma_{c_{i,j}} = \sigma_{i+1,j} - \sigma_{i,j}$

$$l_{c_{i,j}} = \sqrt{(x_{i+1,j} - x_{i,j})^2 + (y_{i+1,j} - y_{i,j})^2 + (z_{i+1,j} - z_{i,j})^2}$$

The resultant velocities and pressure coefficients on the barrier are calculated at a finite number of points called barrier boundary points, whose coordinates in the barrier coordinate system are evaluated by specifying  $(M + 1)$   $r$ 's and  $(N + 1)$   $\theta$ 's.

$$\left. \begin{aligned} \xi_{b_{i,j}} &= \frac{r_{i+1}}{2} [\cos \theta_j + \cos \theta_{j+1}] \\ \eta_{b_{i,j}} &= \frac{r_{i+1}}{2} [\sin \theta_j + \sin \theta_{j+1}] \end{aligned} \right\}, \quad \begin{aligned} i &= 1, \dots, M-1 \\ j &= 1, \dots, N \end{aligned} \quad (51)$$

To transform these coordinates to the reference coordinate system, it is necessary to calculate the direction cosines of the barrier coordinate system.

$$\begin{array}{ccc} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{array}$$

where  $a_{31} = n_{xb}$ ,  $a_{32} = n_{yb}$ ,  $a_{33} = n_{zb}$

$$a_{11} = \frac{n_{zb}}{\sqrt{n_{xb}^2 + n_{zb}^2}}, \quad a_{12} = 0, \quad a_{13} = \frac{-n_{xb}}{\sqrt{n_{xb}^2 + n_{zb}^2}}$$

$$a_{21} = \frac{a_{32}a_{13} - a_{33}a_{12}}{A}, \quad a_{22} = \frac{a_{33}a_{11} - a_{31}a_{13}}{A},$$

$$a_{23} = \frac{a_{31}a_{12} - a_{32}a_{11}}{A}$$

where  $\begin{matrix} n_{xb} \\ n_{yb} \\ n_{zb} \end{matrix} \left. \right\}$  = unit normal vector components of the barrier

$$A = \sqrt{(a_{32}a_{13} - a_{33}a_{12})^2 + (a_{33}a_{11} - a_{31}a_{13})^2 + (a_{31}a_{12} - a_{32}a_{11})^2}$$

The transformed barrier boundary points are given by

$$\left. \begin{matrix} x_{b_{i,j}} = x_{bc} + a_{11} \xi_{b_{i,j}} + a_{21} \eta_{b_{i,j}} \\ y_{b_{i,j}} = y_{bc} + a_{12} \xi_{b_{i,j}} + a_{22} \eta_{b_{i,j}} \\ z_{b_{i,j}} = z_{bc} + a_{13} \xi_{b_{i,j}} + a_{23} \eta_{b_{i,j}} \end{matrix} \right\}, \begin{matrix} i=1, \dots, M-1 \\ j=1, \dots, N \end{matrix} \quad (52)$$

where  $\begin{matrix} x_{bc} \\ y_{bc} \\ z_{bc} \end{matrix} \left. \right\}$  = coordinates of the center of the lift fan in the reference coordinate system

The resultant-induced velocities at the barrier boundary points are found by interpolation.

$$\left. \begin{matrix} v_{xp_{i,j}} = \frac{v_{x_{i,j}} \ell_{2_{i,j}} + v_{x_{i+1,j}} \ell_{1_{i,j}}}{\ell_{1_{i,j}} + \ell_{2_{i,j}}} \\ v_{yp_{i,j}} = \frac{v_{y_{i,j}} \ell_{2_{i,j}} + v_{y_{i+1,j}} \ell_{1_{i,j}}}{\ell_{1_{i,j}} + \ell_{2_{i,j}}} \\ v_{zp_{i,j}} = \frac{v_{z_{i,j}} \ell_{2_{i,j}} + v_{z_{i+1,j}} \ell_{1_{i,j}}}{\ell_{1_{i,j}} + \ell_{2_{i,j}}} \end{matrix} \right\}, \begin{matrix} i=1, \dots, M-1 \\ j=1, \dots, N \end{matrix} \quad (53)$$

where

$$\begin{aligned} l_{1,i,j} &= \sqrt{(x_{b_{i,j}} - x_{i,j})^2 + (y_{b_{i,j}} - y_{i,j})^2 + (z_{b_{i,j}} - z_{i,j})^2} \\ l_{2,i,j} &= \sqrt{(x_{b_{i,j}} - x_{i+1,j})^2 + (y_{b_{i,j}} - y_{i+1,j})^2 + (z_{b_{i,j}} - z_{i+1,j})^2} \end{aligned}$$

The resultant velocity components are composed of two contributions: the resultant-induced velocity components and the velocity components due to one-half the velocity of discontinuity, which are a function of the circumferential and radial velocity components. The components due to the latter contribution are given by

$$\left. \begin{aligned} v_{xd_{i,j}} &= a_{11} v_{d1_{i,j}} + a_{21} v_{d2_{i,j}} \\ v_{yd_{i,j}} &= a_{12} v_{d1_{i,j}} + a_{22} v_{d2_{i,j}} \\ v_{zd_{i,j}} &= a_{13} v_{d1_{i,j}} + a_{23} v_{d2_{i,j}} \end{aligned} \right\}, \begin{array}{l} i = 1, \dots, M-1 \\ j = 1, \dots, N \end{array} \quad (54)$$

where

$$\begin{aligned} v_{d1_{i,j}} &= 0.5 \left( v_{r_{i,j}} \cdot \cos \frac{\theta_j + \theta_{j+1}}{2} - v_{ca_{i,j}} \cdot \sin \frac{\theta_j + \theta_{j+1}}{2} \right) \\ v_{d2_{i,j}} &= 0.5 \left( v_{r_{i,j}} \cdot \sin \frac{\theta_j + \theta_{j+1}}{2} - v_{ca_{i,j}} \cdot \cos \frac{\theta_j + \theta_{j+1}}{2} \right) \end{aligned}$$

Finally, the resultant velocity components are

$$\begin{aligned} v_{xb_{i,j}} &= v_{xp_{i,j}} + v_{xd_{i,j}} \\ v_{yb_{i,j}} &= v_{yp_{i,j}} + v_{yd_{i,j}} \\ v_{zb_{i,j}} &= v_{zp_{i,j}} + v_{zd_{i,j}} \end{aligned} \quad (55)$$

The resultant velocities and pressure coefficients at the barrier boundary points are given by

$$V_{r_{i,j}} = \sqrt{V_{xb_{i,j}}^2 + V_{yb_{i,j}}^2 + V_{zb_{i,j}}^2} \quad (56)$$

$$C_{pb_{i,j}} = 1 - V_{b_{i,j}}^2 \quad (57)$$

In order to calculate the forces and moments on the lift fan, barrier areas are calculated.

$$A_{r_{i,j}} = \left( l_{1_{i,j}} + l_{2_{i,j}} \right) \cdot \left( \theta_{j+1} - \theta_j \right) \cdot \left( \frac{r_{a_{i+1}} + r_{a_i}}{2} \right), \begin{matrix} i=1, \dots, M-1 \\ j=1, \dots, N \end{matrix} \quad (58)$$

$$\text{where } r_{a_1} = \frac{r_i + r_{i+1}}{2}, i = 2, \dots, M-1$$

$$\text{and } r_{a_1} = r_1$$

$$r_{a_M} = r_{M-1}$$

CONSUR's final function is to create a data file containing the barrier boundary-point coordinates, resultant velocity components, pressure coefficients, and barrier areas. This data file is used by subroutine FORCES to compute the force and moment coefficients on the lift fans of the model.

#### USAGE:

COMMON (See program MAIN for blank COMMON description.)

COMMON /JOM2/ NETCS(50), TYPE(10), CPC(10),  
CPE(10), NR(10), NTHETA(10), XBC(10), YBC(10),  
ZBC(10), ANBX(10), ANBY(10), ANBZ(10), ANEX(10),  
ANEY(10), ANEZ(10), H(10), DC(10), SEQNET(7,10),  
R(20,10), THETA(50,10)

CALL CONSUR

Input: ICOM(3) = force and moment option code  
 ICOM(7) = number of lift fans in the model  
 NTSOUT = output data file number  
 NT16 = data file number of the quantities required to compute the force and moment coefficients  
 NT17 = data file number of the quantities required to compute the velocities and pressure coefficients on the lift-fan barriers  
 NETCS = array of values specifying the number of quadrilateral and multihorse-shoe vortex networks used to represent the barrier of each lift fan  
 XBC } = arrays of coordinates of the lift-fan centers in the reference coordinate system  
 YBC }  
 ZBC }  
 ANBX } = arrays of unit normal vector components of the lift-fan barriers  
 ANBY }  
 ANBZ }  
 SEQNET = array of values to identify the type of singularity representing each barrier network. A positive number denotes a quadrilateral vortex network, and a negative number denotes a multihorseshoe vortex network. For example, SEQNET(3) = 5.0 means that quadrilateral vortex network number 5 is used to represent the third singularity network of the lift-fan barrier. Each lift fan has an array of these sequential singularity network identifiers.  
 TYPE = array of values indicating the type of lift fans in the model  
       = 1., model requires half fan  
       = 2., model requires complete fan  
 NR = array of values specifying the number of radial lines used to define the barrier areas of the lift fans

**NTHETA** = array of values specifying the number of circumferential lines used to determine the barrier areas of the lift fans

**R** = array of radial distances,  $r_i$ , used to form the barrier areas. An array of values is required for each lift fan.

**THETA** = array of circumferential angles,  $\theta_j$ , used to form the barrier areas. An array of values is required for each lift fan.

**SUBPROGRAMS**

**CALLED:** None

**ERROR RETURNS:** None

**RESTRICTIONS:** A maximum of ten lift fans in a configuration is allowed. Each fan can have a maximum of 20 r's and 50  $\theta$ 's.

**SUBJECT:** FORTRAN IV Subroutine CTRL

**PURPOSE:** To print an image of the data card, to interpret data commands, and to control flow through the potential-flow program

**METHOD:** The routine first calls subroutine DATA, which writes images of the data cards on the output data file.

Then function INTURP is used to read a data card, to write its contents on the output file, and to see whether it is a control card. A control card has the words GEOM, SOUR, QUAD, MULT, OFF-, OFFb, ENDb, AERO, or EXIT in columns 1 through 4 (b represents a blank). The contents of the remaining columns are written on the output file but are otherwise not used. If the data card is a control card, the appropriate action is taken; otherwise, successive cards are read until a control card is found (this feature permits the insertion of comment cards or blank cards immediately before any control card). The GEOM control card results in a call to subroutine GEOM. On return, data cards are again read until a control card is encountered.

When an AERO control card is read, CTRL calls, in order, subroutines AERO, MATRIX, FLOW, PLOT, and STREAM. An EXIT control card results in termination of the computer run.

If any subroutine detects an error, subroutine DATA is called, which positions the input data file just in front of the next GEOM or EXIT control card.

**USAGE:** COMMON (See program MAIN for blank COMMON description.)

**CALL CTRL**

Input: NTSIN = input data file number  
NTSOUT = output data file number  
ICOM(8) = number of streamlines desired

**SUBPROGRAMS**

**CALLED:**

**DATA  
INTURP  
GEOM  
AERO  
MATRIX  
FLOW  
STREAM  
PLOT**

**ERROR RETURNS:** Subroutine error codes are returned from the subroutine in which they occur.

**RESTRICTIONS:** None

**SUBJECT:** FORTRAN IV Subroutine DATA

**PURPOSE:** To print out input data card images

**METHOD:** The input data file is searched until a GEOM or an EXIT control card is found. If a GEOM card is found, all data cards for the following configuration are printed, and then the input data file is backspaced until it is positioned just in front of the GEOM card. When an EXIT card is encountered, the computer run is terminated.

**USAGE:** COMMON (See program MAIN for blank COMMON description.)

**CALL DATA**

Input: COM(1) } = date of run  
COM(2) }

NTSIN = input data file number

NTSOUT = output data file number

Output: ICOM(6) = data case number

**SUBPROGRAMS CALLED:** None

**ERROR RETURNS:** None

**RESTRICTIONS:** If the EXIT card does not appear in the data deck, the computer run will be terminated by the End of File (EOF) on the input data file.

**SUBJECT:** FORTRAN IV Subroutine DATE

**PURPOSE:** To obtain the current date

**METHOD:** This is a dummy subroutine that returns blanks. An appropriate change may be made at the user's installation to return the true date.

**USAGE:** CALL DATE (A, B)

Output: A = first portion of the date (alphabetic)  
B = last portion of the date (alphabetic)

**SUBPROGRAMS CALLED:** None

**ERROR RETURNS:** None

**RESTRICTIONS:** None

**SUBJECT:** FORTRAN IV Subroutine ENDSL

**PURPOSE:** To print a message stating the reason for termination of a streamline

**METHOD:** There are six possible ways for a streamline to terminate:

1. The number of points defining the streamline exceeds 50.
2. The streamline exceeded the maximum allowable length (the maximum length is specified by the user).
3. The streamline failed to enter the starting source panel.
4. The streamline reached a near-stagnation region.
5. The periphery search of the source-panel networks for an entrance was unsuccessful.
6. The source-panel-defining quantities were incorrectly read from a data file.

If a streamline is terminated because it reached a near-stagnation region, the boundary point and the velocity at the boundary point of the last panel traversed and the panel the streamline failed to enter are printed.

**USAGE:**

COMMON (See program MAIN for blank COMMON description.)

COMMON /SL2/ (See subroutine PANEL.)

CALL ENDSL (ITERM, AL, IP, JP, KP, LP, I, J, K, L)

Input: NTSOUT = output data file number

ITERM = 1, number of streamline points exceeds 50

= 2, streamline exceeded maximum specified length

= 3, streamline failed to enter starting panel

= 4, streamline reached a near-stagnation region

- = 5, network periphery search was unsuccessful
- = 6, source-panel-defining quantities were incorrectly read from a data file

<b>AL</b>	= specified maximum length of a streamline
<b>IP</b>	
<b>JP</b>	
<b>KP</b>	
<b>LP</b>	
	}
<b>I</b>	
<b>J</b>	
<b>K</b>	
<b>L</b>	
	}
<b>XBP</b>	
<b>YBP</b>	
<b>ZBP</b>	
	}
<b>VFX</b>	
<b>VFY</b>	
<b>VFZ</b>	
	}

= indices of the last source panel traversed by the streamline (the  $IP^{th}$  panel in the  $JP^{th}$  column of the  $K^{th}$  network;  $LP$  is the panel's accumulative number)

= indices of the source panel the streamline failed to enter (the  $I^{th}$  panel in the  $J^{th}$  column of the  $K^{th}$  network;  $L$  is the panel's accumulative number)

= arrays of the coordinates of the source-panel boundary points

= arrays of the velocity components, at the source-panel boundary points

**SUBPROGRAMS**

**CALLED:** None

**ERROR RETURNS:** None

**RESTRICTIONS:** None

**SUBJECT:** FORTRAN IV Subroutine FAIL  
**PURPOSE:** To determine the cause for the failure of a streamline to enter a panel  
**METHOD:** This routine tests the streamline entrance angle of the previous panel (last panel the streamline successfully traversed) and the current panel (panel streamline failed to enter) to determine whether the streamline has a rapid change in direction; i. e., reached a near-stagnation region. The streamline reached this region if

$$\cos \theta_p < \epsilon_{11} \quad (59)$$

or

$$\cos \theta < \epsilon_{11} \quad (60)$$

where  $\theta_p$  = the streamline entrance angle of the previous panel

$\theta$  = the streamline entrance angle of the current panel

$\epsilon_{11}$  =  $\cos 20^\circ$ , cosine of the limiting entrance angle

The cosine of the streamline entrance angle of the current panel is defined as

$$\cos \theta = \frac{|\hat{\mathbf{V}} \cdot \hat{\mathbf{d}}_e|}{\left( \sqrt{V_\xi^2 + V_\eta^2} \right) \sqrt{(\xi_{e+1} - \xi_e)^2 + (\eta_{e+1} - \eta_e)^2}} \quad (61)$$

where  $\hat{\mathbf{V}} \cdot \hat{\mathbf{d}}_e = V_\xi \cdot (\xi_{e+1} - \xi_e) + V_\eta \cdot (\eta_{e+1} - \eta_e)$

$V_\xi \quad \left. \right\}$  = the velocity components in the element  
 $V_\eta \quad \left. \right\}$  coordinate system

$\xi_e \quad \left. \right\}$  corner-point coordinates of the source  
 $\eta_e \quad \left. \right\}$  = panel in the element coordinate system.  
 The subscript e is the corner-point number.

The streamline entrance angle and  $(\hat{\mathbf{V}} \cdot \hat{\mathbf{d}}_e)$  of the previous panel are calculated in subroutine STREAM. If the two stagnation tests fail, but the sign of the previous panel is the same as the sign of the current panel, the streamline is terminated.

If the streamline did not terminate, one additional test is performed that determines a new exit side and the fractional distance of the streamline-starting point along the side ( $d/d_e$ ) of the current panel. If the sign of  $\hat{V} \cdot \hat{d}_e$  is positive, the exit side and fractional-starting distance are given by

$$e = e - 1 \quad (62)$$

$$d/d_e = 0.99 \quad (63)$$

However, if the sign of  $\hat{V} \cdot \hat{d}_e$  is negative, the exit side and fractional-starting distance are given by

$$e = e + 1 \quad (64)$$

$$d/d_e = 0.01 \quad (65)$$

Subroutine STREAM uses the new values of  $e$  and  $d/d_e$  to continue traversing the panels.

#### USAGE:

COMMON /SL2/ (See subroutine PANEL.)

INTEGER EE, EX

CALL FAIL (ISL, L, LP, EE, EX, DDX, ITERM)

Input:  $\left. \begin{matrix} VG \\ VH \end{matrix} \right\}$  = velocity components of the current panel in the element coordinate system

VDEP = the value of  $\hat{V} \cdot \hat{d}_e$  for the previous panel

COSTP = cosine of the streamline entrance angle of the previous panel,  $\cos \theta_p$

E11 = limiting entrance angle,  $\epsilon_{11}$

ISL = index for the number of points in a streamline

L = accumulative source-panel number of the current panel

LP = accumulative source-panel number of the previous panel

EE = entrance side number of the current panel

**Output:** EX = exit side number of the current panel  
**DDX** = fractional-starting distance of the streamline along the exit side of the current panel  
**ITERM** = 0, the streamline did not terminate  
= 3, the streamline failed to enter the starting panel (ISL = 1)  
= 4, the streamline reached a near-stagnation region

**SUBPROGRAMS  
CALLED:** None

**ERROR RETURNS:** None

**RESTRICTIONS:** The previous and current panels must be four-sided.

**SUBJECT:**

**FORTRAN IV Subroutine FLOW**

**PURPOSE:**

To calculate the velocity and pressure coefficient at the boundary points on the potential-flow model and at off-body points

**METHOD:**

This routine has five primary functions:

1. Calculate and print the resultant velocity components, velocity magnitude, pressure coefficient, singularity strength, and resultant normal velocity at each singularity boundary point.
2. Calculate and print the resultant velocity components, velocity magnitude, and pressure coefficient at each off-body point.
3. Create a data file containing the boundary-point coordinates, unit normal vector components, areas, pressure coefficients, and singularity strengths of the source-panel singularities. This data file is used by subroutine FORCES to compute the force and moment coefficients of the model represented by source panels.
4. Create a data file containing the boundary-point coordinates, resultant velocity components, and singularity strengths of the lift-fan barrier singularities. This data file is used by subroutine CONSUR to compute the lift-fan barrier velocities and pressures.
5. Create a data file containing the title, number of source-panel networks, network dimensions, number of solutions, angles of attack, angles of yaw, and boundary-point coordinates, velocities, and pressure coefficients of the source-panel singularities. This data file is used by subroutine PLOT to plot the velocity and pressure distributions on the model.

Subroutine FLOW reads the reference quantities used to define the force and moment coefficients from the input data file; then the lift-fan barrier geometry is read and stored in a labeled COMMON block for use by subroutine CONSUR.

At the beginning of the computations for a particular solution, a short title block is printed that contains the solution number and the type of solution desired—angle of attack/angle of yaw specification or a zero free-stream velocity specification. Then, using subroutine READTP,

the resultant-induced velocity components ( $u$ ,  $v$ ,  $w$ ) at the boundary points and off-body points are read from data files created by the matrix routines. A resultant-induced velocity is the velocity at a point due to all the singularities of the model. The resultant velocity at a point is the sum of the resultant-induced velocity and the free-stream velocity at that point. Its components are

$$\begin{aligned} V_x &= u + V_{\infty x} \\ V_y &= v + V_{\infty y} \\ V_z &= w + V_{\infty z} \end{aligned} \quad (66)$$

The magnitude of velocity and the pressure coefficient are obtained from

$$V = \sqrt{V_x^2 + V_y^2 + V_z^2} \quad (67)$$

and

$$C_p = 1 - V^2 \quad (68)$$

If the point is a singularity boundary point, the resultant normal velocity is

$$V_{RN} = V_x n_x + V_y n_y + V_z n_z \quad (69)$$

where  $\left. \begin{matrix} n_x \\ n_y \\ n_z \end{matrix} \right\}$  = unit normal vector components at the boundary point

This velocity indicates the validity of the solution (Note: a velocity in the range  $10^{-8}$  to  $10^{-16}$  is caused by computer roundoff). If the user specifies a normal velocity (see subroutine SNF), it, plus any roundoff, will appear here.

The singularity strengths are read from another data file created by the matrix routines.

All geometric quantities (boundary-point coordinates, unit normal vector components, singularity areas, and off-body point coordinates) are read from the boundary-point quantities data file and stored in the array BP.

**USAGE:**

**COMMON** (See program **MAIN** for blank **COMMON** description.)

**CALL FLOW**

Input:    **ICOM(1)** = 1, symmetric flow  
              = 3, unsymmetric flow  
**ICOM(2)** = maximum matrix partition size  
**ICOM(3)** = force and moment option code  
**ICOM(4)** = plot option code  
**ICOM(7)** = number of lift fans in the model  
**NT1**      = data file number of the singularity strengths  
**NT2**      = data file number of the x component of the resultant-induced velocities, u  
**NT3**      = data file number of the y component of the resultant-induced velocities, v  
**NT4**      = data file number of the z component of the resultant-induced velocities, w  
**NTSIN**    = input data file number  
**NTSOUT**   = output data file number  
**NT8**      = data file number of the boundary-point quantities  
**NTPLOT**   = data file number of the results to be plotted  
**NT16**     = data file number of the quantities required to compute the force and moment coefficients  
**NT17**     = data file number of the quantities required to compute the velocities and pressures on the lift-fan barrier  
**NPR**      = number of row partitions in the aerodynamic solution matrices

**LS** = array containing the number of row elements (resultant-induced velocities or singularity strengths) in each row partition  
**MATSP** = number of source-panel singularities  
**MATQV** = number of quadrilateral vortex singularities  
**MATMHV** = number of multihorseshoe vortex singularities  
**NETSP** = number of source-panel networks  
**MSP } NSP }** = arrays containing the dimensions of the source-panel networks  
**NETQV** = number of quadrilateral vortex networks  
**MQV } NQV }** = arrays containing the dimensions of the quadrilateral vortex networks  
**NETMHV** = number of multihorseshoe vortex networks  
**NHV** = array containing the dimensions of the multihorseshoe vortex networks  
**NRHS** = number of simultaneous solutions  
**ALPHA** = array of angles of attack,  $\alpha$   
**PSI** = array of angles of yaw,  $\psi$   
**FSVX** = array of the x component of the free-stream velocity,  $V_{\infty x}$   
**FSVY** = array of the y component of the free-stream velocity,  $V_{\infty y}$   
**FSVZ** = array of the z component of the free-stream velocity,  $V_{\infty z}$

**SUBPROGRAMS CALLED:** CONSUR  
**FORCES**  
**READTP**  
**ERROR RETURNS:** None  
**RESTRICTIONS:** None

SUBJECT: FORTRAN IV Subroutine FORCES

PURPOSE: To calculate the force and moment coefficients on the potential-flow model at a given angle of attack and angle of yaw

METHOD: This routine is divided into three sections:

- Calculation of the force and moment coefficients on the source-paneled model,
- Calculation of the force and moment coefficients on the lift fans, and
- Calculation of the total force and moment coefficients on the potential-flow model.

1. Force and Moment Coefficients of the Source-Paneled Model

The nomenclature used in this section is listed below.

$x_r \quad \left. \begin{matrix} \\ y_r \\ z_r \end{matrix} \right\}$  = moment reference coordinates

$c_r$  = reference chord

$b_r$  = reference span

$S_r$  = reference planform area

$\alpha$  = angle of attack

$\psi$  = angle of yaw

NET = total number of source-panel networks

$N_k$  = number of columns in the  $k^{\text{th}}$  network

$M_{jk}$  = number of panels in the  $j^{\text{th}}$  column of the  $k^{\text{th}}$  network

$x_i \quad \left. \begin{matrix} \\ y_i \\ z_i \end{matrix} \right\}$  = boundary-point coordinates of the  $i^{\text{th}}$  panel in the  $j^{\text{th}}$  column of the  $k^{\text{th}}$  network

$n_{x_i} \quad \left. \begin{matrix} \\ n_{y_i} \\ n_{z_i} \end{matrix} \right\}$  = unit normal vector components of the  $i^{\text{th}}$  panel

$A_i$  = area of the  $i^{\text{th}}$  panel

$C_{p_i}$  = pressure coefficient at the boundary point of the  $i^{\text{th}}$  panel

$\sigma_i$  = singularity strength of the  $i^{\text{th}}$  panel

$r_{x_i} = x_i - x_r$

$r_{y_i} = y_i - y_r$

$r_{z_i} = z_i - z_r$

The symmetric and unsymmetric flow formulation of the calculations follows.

a. Symmetric Flow

The force coefficients per source-panel column are given by

$$C_{Fx_{jk}} = \frac{1}{S_r} \sum_{i=1}^{M_{jk}} C_{p_i} A_i n_{x_i} \quad (70)$$

$$C_{Fz_{jk}} = \frac{1}{S_r} \sum_{i=1}^{M_{jk}} C_{p_i} A_i n_{z_i}$$

The total force coefficients due to all the source panels are

$$C_{Fxs} = 2 \sum_{k=1}^{\text{NET}} \sum_{j=1}^{N_k} C_{Fx_{jk}} \quad (71)$$

$$C_{Fzs} = 2 \sum_{k=1}^{\text{NET}} \sum_{j=1}^{N_k} C_{Fz_{jk}}$$

The moment coefficient per source-panel column is given by

$$C_{my_{jk}} = -\frac{1}{c_r S_r} \sum_{i=1}^{M_{jk}} C_{p_i} A_i [n_{x_i} r_{z_i} - n_{z_i} r_{x_i}] \quad (72)$$

The total moment coefficient due to all the source panels is

$$C_{mys} = 2 \sum_{k=1}^{NET} \sum_{j=1}^{N_k} C_{my_{jk}} \quad (73)$$

The lift and drag coefficients per source-panel column are given by

$$C_{L_{jk}} = -C_{Fx_{jk}} \sin \alpha + C_{Fz_{jk}} \cos \alpha \quad (74)$$

$$C_{D_{jk}} = C_{Fx_{jk}} \cos \alpha + C_{Fz_{jk}} \sin \alpha \quad (75)$$

The total lift and drag coefficients due to all the source panels are

$$C_{Ls} = 2 \sum_{k=1}^{NET} \sum_{j=1}^{N_k} C_{L_{jk}} \quad (76)$$

$$C_{Ds} = 2 \sum_{k=1}^{NET} \sum_{j=1}^{N_k} C_{D_{jk}} \quad (77)$$

The net source-panel strength per source-panel column is given by

$$NS_{jk} = \sum_{i=1}^{M_{jk}} \sigma_i A_i \quad (78)$$

The total strength due to all the source panels is

$$NS = 2 \sum_{k=1}^{NET} \sum_{j=1}^{N_k} NS_{jk} \quad (79)$$

The wetted area per source-panel column is given by

$$A_{wjk} = \sum_{i=1}^{M_{jk}} A_i \quad (80)$$

and the total wetted area of the source-panel model is

$$A_{ws} = \sum_{k=1}^{\text{NET}} \sum_{j=1}^{N_k} A_{wjk} \quad (81)$$

#### b. Unsymmetric Flow

The force coefficients per source-panel column are given by

$$\begin{aligned} C_{Fx_{jk}} &= -\frac{1}{S_r} \sum_{i=1}^{M_{jk}} C_{pi} A_i n_{x_i} \\ C_{Fy_{jk}} &= -\frac{1}{S_r} \sum_{i=1}^{M_{jk}} C_{pi} A_i n_{y_i} \\ C_{Fz_{jk}} &= -\frac{1}{S_r} \sum_{i=1}^{M_{jk}} C_{pi} A_i n_{z_i} \end{aligned} \quad (82)$$

The total force coefficients due to all the source panels are

$$\begin{aligned} C_{Fxs} &= \sum_{k=1}^{\text{NET}} \sum_{j=1}^{N_k} C_{Fx_{jk}} \\ C_{Fys} &= \sum_{k=1}^{\text{NET}} \sum_{j=1}^{N_k} C_{Fy_{jk}} \\ C_{Fzs} &= \sum_{k=1}^{\text{NET}} \sum_{j=1}^{N_k} C_{Fz_{jk}} \end{aligned} \quad (83)$$

The moment coefficients per source-panel column are given by

$$\begin{aligned}
 C_{mx_{jk}} &= -\frac{1}{b_r s_r} \sum_{i=1}^{M_{jk}} C_{p_i} A_i [n_{z_i} r_{y_i} - n_{y_i} r_{z_i}] \\
 C_{my_{jk}} &= -\frac{1}{c_r s_r} \sum_{i=1}^{M_{jk}} C_{p_i} A_i [n_{x_i} R_{z_i} - n_{z_i} R_{x_i}] \\
 C_{mz_{jk}} &= -\frac{1}{b_r s_r} \sum_{i=1}^{M_{jk}} C_{p_i} A_i [n_{y_i} r_{x_i} - n_{x_i} r_{y_i}]
 \end{aligned} \tag{84}$$

The total moment coefficients due to all the source panels are

$$\begin{aligned}
 C_{mzs} &= \sum_{k=1}^{\text{NET}} \sum_{j=1}^{N_k} C_{mx_{jk}} \\
 C_{mys} &= \sum_{k=1}^{\text{NET}} \sum_{j=1}^{N_k} C_{my_{jk}} \\
 C_{mzs} &= \sum_{k=1}^{\text{NET}} \sum_{j=1}^{N_k} C_{mz_{jk}}
 \end{aligned} \tag{85}$$

The lift, drag, and side force coefficients per source-panel column are given by

$$\begin{aligned}
 C_{L_{jk}} &= -C_{Fx_{jk}} \sin \alpha \cos \psi \\
 &\quad + C_{Fy_{jk}} \sin \alpha \sin \psi + C_{Fz_{jk}} \cos \alpha
 \end{aligned} \tag{86}$$

$$\begin{aligned}
 C_{D_{jk}} &= C_{Fx_{jk}} \cos \alpha \cos \psi \\
 &\quad - C_{Fy_{jk}} \cos \alpha \sin \psi + C_{Fz_{jk}} \sin \alpha
 \end{aligned} \tag{87}$$

$$C_{Y_{jk}} = C_{Fx_{jk}} \sin \psi + C_{Fy_{jk}} \cos \psi \tag{88}$$

The total lift, drag, and side force coefficients due to all the source panels are

$$C_{Ls} = \sum_{k=1}^{NET} \sum_{j=1}^{N_k} C_{Ljk} \quad (89)$$

$$C_{Ds} = \sum_{k=1}^{NET} \sum_{j=1}^{N_k} C_{Djk} \quad (90)$$

$$C_{Ys} = \sum_{k=1}^{NET} \sum_{j=1}^{N_k} C_{Yjk} \quad (91)$$

The net strength per source-panel column is given by

$$NS_{jk} = \sum_{i=1}^{M_{jk}} \sigma_i A_i \quad (92)$$

The total strength due to all the source panels is

$$NS = \sum_{k=1}^{NET} \sum_{j=1}^{N_k} NS_{jk} \quad (93)$$

The wetted area per source-panel column is given by

$$A_{w_{jk}} = \sum_{i=1}^{M_{jk}} A_i \quad (94)$$

And the total wetted area of the source-paneled model is

$$A_{ws} = \sum_{k=1}^{NET} \sum_{j=1}^{N_k} A_{w_{jk}} \quad (95)$$

## 2. Force and Moment Coefficients of the Lift Fans

A lift fan consists of a barrier (fan surface), centerbody, exit plane (flow exiting from the lower surface), and centerbody base, as shown in Figure 70. The barrier and exit plane are called control surfaces.

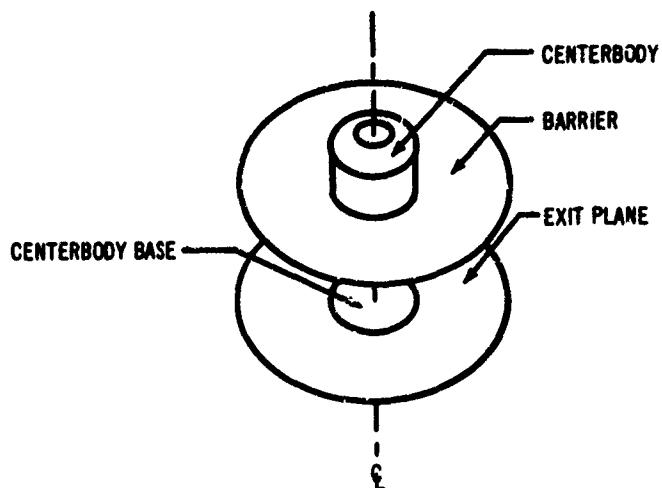


Figure 70. Subroutine FORCES—Lift-Fan Assembly.

The barrier, exit plane, and centerbody base are used in the analysis for force and moment calculations on the lift fan. The centerbody is represented by source panels; hence, the forces and moments on it are calculated in the previous section. A maximum of ten lift fans can be accepted by the program.

The nomenclature used in this section is listed below.

$x_r$	= moment reference coordinates
$y_r$	
$z_r$	
$c_r$	= reference chord
$b_r$	= reference span
$S_r$	= reference planform area
$\alpha$	= angle of attack
$\psi$	= angle of yaw
$N_k$	= number of columns in the barrier of the $k^{\text{th}}$ lift fan

$M_{jk}$  = number of barrier areas in the  $j^{\text{th}}$  column of the  $k^{\text{th}}$  lift fan

$x_i$   
 $y_i$   
 $z_i$

= boundary-point coordinates of the  $i^{\text{th}}$  barrier area in the  $j^{\text{th}}$  column of the  $k^{\text{th}}$  lift fan

$A_{b_i}$  =  $i^{\text{th}}$  barrier area

$C_{p_{b_i}}$  = pressure coefficient at the boundary point of the  $i^{\text{th}}$  barrier area

$V_{xb_i}$   
 $V_{yb_i}$   
 $V_{zb_i}$

= velocity components at the boundary point of the  $i^{\text{th}}$  barrier area

$q_{b_i}$  = normal velocity into the  $i^{\text{th}}$  barrier area,  
 $q_{b_i} = -n_{xb_k} V_{xb_i} + n_{yb_k} V_{yb_i} + n_{zb_k} V_{zb_i}$

$r_{x_i}$  =  $x_i - x_r$

$r_{y_i}$  =  $y_i - y_r$

$r_{z_i}$  =  $z_i - z_r$

$x_{b_k}$   
 $y_{b_k}$   
 $z_{b_k}$

= coordinates of the center of the  $k^{\text{th}}$  barrier

$n_{xb_k}$   
 $n_{yb_k}$   
 $n_{zb_k}$

= unit normal vector components of the  $k^{\text{th}}$  barrier

$r_{xb_k}$  =  $x_{b_k} - x_r$

$r_{yb_k}$  =  $y_{b_k} - y_r$

$r_{zb_k}$  =  $z_{b_k} - z_r$

- $$\left. \begin{array}{l} t_{x_k} \\ t_{y_k} \\ t_{z_k} \end{array} \right\} = \text{unit vector components in the direction of the jet efflux}$$
- $$E_k = n_{xb} t_{x_k} + n_{yb} t_{y_k} + n_{zb} t_{z_k}$$
- $$C_{pe_k} = \text{assumed exit pressure coefficient of the } k^{\text{th}} \text{ lift fan}$$
- $$A_{c_k} = \text{area of centerbody base, } = \frac{\pi}{8} d_{c_k}^2, \text{ where } d_{c_k} \text{ is the diameter of the base}$$
- $$C_{pc_k} = \text{assumed centerbody base pressure coefficient of the } k^{\text{th}} \text{ lift fan}$$
- $$h_k = \text{average distance between the barrier and exit plane of the } k^{\text{th}} \text{ fan}$$

The symmetric and unsymmetric flow formulation of the calculations follows. All calculations refer to the  $k^{\text{th}}$  lift fan.

#### a. Symmetric Flow

The pressure force coefficients per column on the barrier are given by

$$C_{Fx1_{jk}} = -\frac{n_{xb}}{S_r} k \sum_{i=1}^{M_{jk}} C_{pb_i} A_{b_i} \quad (96)$$

$$C_{Fz1_{jk}} = -\frac{n_{zb}}{S_r} k \sum_{i=1}^{M_{jk}} C_{pb_i} A_{b_i}$$

The total pressure force coefficients on the barrier are

$$C_{Fx1_k} = 2 \sum_{j=1}^{N_k} C_{Fx1_{jk}} \quad (97)$$

$$C_{Fz1_k} = 2 \sum_{j=1}^{N_k} C_{Fz1_{jk}}$$

The pressure force coefficients per column on the exit plane are given by

$$C_{Fx2jk} = \frac{n_{xb}}{S_r} k C_{p_{e_k}} \sum_{i=1}^{M_{jk}} A_{b_i} \quad (98)$$

$$C_{Fz2jk} = \frac{n_{zb}}{S_r} k C_{p_{e_k}} \sum_{i=1}^{M_{jk}} A_{b_i}$$

The total pressure force coefficients on the exit plane are

$$C_{Fx2_k} = 2 \sum_{j=1}^{N_k} C_{Fx2_{jk}} \quad (99)$$

$$C_{Fz2_k} = 2 \sum_{j=1}^{N_k} C_{Fz2_{jk}}$$

The pressure force coefficients on the centerbody base are given by

$$C_{Fxc_k} = 2 \frac{n_{xb}}{S_r} k C_{p_{c_k}} A_{c_k} \quad (100)$$

$$C_{Fzc_k} = 2 \frac{n_{zb}}{S_r} k C_{p_{c_k}} A_{c_k}$$

The momentum force coefficients per column on the barrier are given by

$$C_{Fx3_{jk}} = \frac{2}{S_r} \sum_{i=1}^{M_{jk}} q_{b_i} V_{xb_i} A_{b_i} \quad (101)$$

$$C_{Fz3_{jk}} = \frac{2}{S_r} \sum_{i=1}^{M_{jk}} q_{b_i} V_{zb_i} A_{b_i}$$

The total momentum force coefficients on the barrier are

$$C_{Fx3_k} = 2 \sum_{j=1}^{N_k} C_{Fx3_{jk}} \quad (102)$$

$$C_{Fz3_k} = 2 \sum_{j=1}^{N_k} C_{Fz3_{jk}}$$

The momentum force coefficients per column on the exit plane are given by

$$C_{Fx4_{jk}} = \frac{2}{S_r} \left( \frac{t_x}{E} \right)_k \sum_{i=1}^{M_{jk}} q_{b_i}^2 A_{b_i} \quad (103)$$

$$C_{Fz4_{jk}} = \frac{2}{S_r} \left( \frac{t_z}{E} \right)_k \sum_{i=1}^{M_{jk}} q_{b_i}^2 A_{b_i}$$

The total momentum force coefficients on the exit plane are

$$C_{Fx4_k} = 2 \sum_{j=1}^{N_k} C_{Fx4_{jk}} \quad (104)$$

$$C_{Fz4_k} = 2 \sum_{j=1}^{N_k} C_{Fz4_{jk}}$$

The pressure moment coefficient per column on the barrier is given by

$$C_{my1_{jk}} = -\frac{c_r}{S_r} \sum_{i=1}^{M_{jk}} C_{p_{b_i}} \left| r_{z_i} n_{x b_k} - r_{x_i} n_{z b_k} \right| A_{b_i} \quad (105)$$

The total pressure moment coefficient on the barrier is

$$C_{my1_k} = 2 \sum_{j=1}^{N_k} C_{my1_{jk}} \quad (106)$$

The pressure moment coefficient per column on the exit plane is given by

$$C_{my2_{jk}} = \frac{1}{c_r S_r} C_{p_{e_k}} \left[ r_{zb_k} n_{xb_k} - r_{xb_k} n_{zb_k} \right] \sum_{i=1}^{M_{jk}} A_{b_i} \quad (107)$$

The total pressure moment coefficient on the exit plane is

$$C_{my2_k} = 2 \sum_{j=1}^{N_k} C_{my2_{jk}} \quad (108)$$

The momentum moment coefficient per column on the barrier is given by

$$C_{my3_{jk}} = \frac{2}{c_r S_r} \sum_{i=1}^{M_{jk}} q_{b_i} \left[ r_{z_i} V_{xb_i} - r_{x_i} V_{zb_i} \right] A_{b_i} \quad (109)$$

The total momentum moment coefficient on the barrier is

$$C_{my3_k} = 2 \sum_{j=1}^{N_k} C_{my3_{jk}} \quad (110)$$

The momentum moment coefficient per column on the exit plane is given by

$$C_{my4_{jk}} = \frac{2}{c_r S_r E} \sum_{i=1}^{M_{jk}} q_{b_i}^2 \left[ (r_{z_i} - n_{zb_k} h_k) t_{x_k} - (r_{x_i} - n_{xb_k} h_k) t_{z_k} \right] A_{b_i} \quad (111)$$

The total momentum moment coefficient on the exit plane is

$$C_{my4_k} = 2 \sum_{j=1}^{N_k} C_{my4_{jk}} \quad (112)$$

The pressure moment coefficient on the centerbody base is given by

$$C_{myc_k} = \frac{2}{c_r S_r} C_{pc_k} [r_{zb_k} n_{xb_k} - r_{xb_k} n_{zb_k}] A_{c_k} \quad (113)$$

The lift and drag coefficients per column on the barrier and the exit plane are given by

$$C_{L\ell_{jk}} = -C_{Fx\ell_{jk}} \sin \alpha + C_{Fz\ell_{jk}} \cos \alpha \quad (114)$$

$$C_{D\ell_{jk}} = C_{Fx\ell_{jk}} \cos \alpha + C_{Fz\ell_{jk}} \sin \alpha \quad (115)$$

where  $\ell = 1$ , pressure force on barrier

$\ell = 2$ , pressure force on exit plane

$\ell = 3$ , momentum force on barrier

$\ell = 4$ , momentum force on exit plane

$\ell = c$ , pressure force on centerbody base

The total lift and drag coefficients on the barrier and exit plane are

$$C_{L\ell_k} = 2 \sum_{j=1}^{N_k} C_{L\ell_{jk}} \quad (116)$$

$$C_{D\ell_k} = 2 \sum_{j=1}^{N_k} C_{D\ell_{jk}} \quad (117)$$

where  $\ell = 1, 2, 3, 4$ , and  $c$ , as before

The wetted area per column of the barrier and exit plane is given by

$$A_{wb_{jk}} = \sum_{i=1}^{M_{jk}} A_{b_i} \quad (118)$$

$$A_{we_{jk}} = A_{wb_{jk}} \quad (119)$$

The total wetted area on the barrier and exit plane is

$$A_{wb_k} = 2 \sum_{j=1}^{N_k} A_{wb_{jk}} \quad (120)$$

$$A_{we_k} = A_{wb_k} \quad (121)$$

### b. Unsymmetric Flow

The pressure force coefficients per column on the barrier are given by

$$\begin{aligned} C_{Fx1_{jk}} &= \frac{n_{xb}}{S_r} k \sum_{i=1}^{M_{jk}} C_{pb_i} A_{b_i} \\ C_{Fy1_{jk}} &= \frac{n_{yb}}{S_r} k \sum_{i=1}^{M_{jk}} C_{pb_i} A_{b_i} \\ C_{Fz1_{jk}} &= \frac{n_{zb}}{S_r} k \sum_{i=1}^{M_{jk}} C_{pb_i} A_{b_i} \end{aligned} \quad (122)$$

The total pressure force coefficients on the barrier are

$$\begin{aligned} C_{Fx1_k} &= \sum_{j=1}^{N_k} C_{Fx1_{jk}} \\ C_{Fy1_k} &= \sum_{j=1}^{N_k} C_{Fy1_{jk}} \\ C_{Fz1_k} &= \sum_{j=1}^{N_k} C_{Fz1_{jk}} \end{aligned} \quad (123)$$

The pressure force coefficients per column on the exit plane are given by

$$\begin{aligned} C_{Fx2_{jk}} &= \frac{n_{xb}}{S_r} k C_{pe_k} \sum_{i=1}^{M_{jk}} A_{b_i} \\ C_{Fy2_{jk}} &= \frac{n_{yb}}{S_r} k C_{pe_k} \sum_{i=1}^{M_{jk}} A_{b_i} \\ C_{Fz2_{jk}} &= \frac{n_{zb}}{S_r} k C_{pe_k} \sum_{i=1}^{M_{jk}} A_{b_i} \end{aligned} \quad (124)$$

The total pressure force coefficients on the exit plane are

$$\begin{aligned} C_{Fx2_k} &= \sum_{j=1}^{N_k} C_{Fx2_{jk}} \\ C_{Fy2_k} &= \sum_{j=1}^{N_k} C_{Fy2_{jk}} \\ C_{Fz2_k} &= \sum_{j=1}^{N_k} C_{Fz2_{jk}} \end{aligned} \quad (125)$$

The pressure force coefficients on the centerbody base are given by

$$\begin{aligned} C_{Fx0_k} &= \frac{n_{xb}}{S_r} k C_{p_{c_k}} A_{c_k} \\ C_{Fyc_k} &= \frac{n_{yb}}{S_r} k C_{p_{c_k}} A_{c_k} \quad (126) \\ C_{Fzc_k} &= \frac{n_{zb}}{S_r} k C_{p_{c_k}} A_{c_k} \end{aligned}$$

The momentum force coefficients per column on the barrier are given by

$$\begin{aligned} C_{Fx3_{jk}} &= \frac{2}{S_r} \sum_{i=1}^{M_{jk}} q_{b_i} V_{xb_i} A_{b_i} \\ C_{Fy3_{jk}} &= \frac{2}{S_r} \sum_{i=1}^{M_{jk}} q_{b_i} V_{yb_i} A_{b_i} \quad (127) \\ C_{Fz3_{jk}} &= \frac{2}{S_r} \sum_{i=1}^{M_{jk}} q_{b_i} V_{zb_i} A_{b_i} \end{aligned}$$

The total momentum force coefficients on the barrier are

$$\begin{aligned} C_{Fx3_k} &= \sum_{j=1}^{N_k} C_{Fx3_{jk}} \\ C_{Fy3_k} &= \sum_{j=1}^{N_k} C_{Fy3_{jk}} \quad (128) \\ C_{Fz3_k} &= \sum_{j=1}^{N_k} C_{Fz3_{jk}} \end{aligned}$$

The momentum force coefficients per column on the exit plane are given by

$$C_{Fx4_{jk}} = \frac{2}{S_r} \frac{t_x}{E} k \sum_{i=1}^{M_{jk}} q_{b_i}^2 A_{b_i}$$

$$C_{Fy4_{jk}} = \frac{2}{S_r} \frac{t_y}{E} k \sum_{i=1}^{M_{jk}} q_{b_i}^2 A_{b_i} \quad (129)$$

$$C_{Fz4_{jk}} = \frac{2}{S_r} \frac{t_z}{E} k \sum_{i=1}^{M_{jk}} q_{b_i}^2 A_{b_i}$$

The total momentum force coefficients on the exit plane are

$$C_{Fx4_k} = \sum_{j=1}^{N_k} C_{Fx4_{jk}}$$

$$C_{Fy4_k} = \sum_{j=1}^{N_k} C_{Fy4_{jk}} \quad (130)$$

$$C_{Fz4_k} = \sum_{j=1}^{N_k} C_{Fz4_{jk}}$$

The pressure moment coefficients per column on the barrier are given by

$$C_{mx1_{jk}} = -\frac{1}{b_r S_r} \sum_{i=1}^{M_{jk}} C_{p_{b_i}} [r_{y_i} n_{zb_k} - r_{z_i} n_{yb_k}] A_{b_i}$$

$$C_{my1_{jk}} = -\frac{1}{c_r S_r} \sum_{i=1}^{M_{jk}} C_{p_{b_i}} [r_{z_i} n_{xb_k} - r_{x_i} n_{zb_k}] A_{b_i} \quad (131)$$

$$C_{mz1_{jk}} = -\frac{1}{b_r S_r} \sum_{i=1}^{M_{jk}} C_{p_{b_i}} [r_{x_i} n_{yb_k} - r_{y_i} n_{xb_k}] A_{b_i}$$

The total pressure moment coefficients on the barrier are

$$\begin{aligned} C_{mx1_k} &= \sum_{j=1}^{N_k} C_{mx1_{jk}} \\ C_{my1_k} &= \sum_{j=1}^{N_k} C_{my1_{jk}} \\ C_{mz1_k} &= \sum_{j=1}^{N_k} C_{mz1_{jk}} \end{aligned} \quad (132)$$

The pressure moment coefficients per column on the exit plane are given by

$$\begin{aligned} C_{mx2_{jk}} &= \frac{1}{b_r S_r} C_{p_{e_k}} [r_{yb_k} n_{zb_k} - r_{zb_k} n_{yb_k}] \sum_{i=1}^{M_{jk}} A_{b_i} \\ C_{my2_{jk}} &= \frac{1}{c_r S_r} C_{p_{e_k}} [r_{zb_k} n_{xb_k} - r_{xb_k} n_{zb_k}] \sum_{i=1}^{M_{jk}} A_{b_i} \\ C_{mz2_{jk}} &= \frac{1}{b_r S_r} C_{p_{e_k}} [r_{xb_k} n_{yb_k} - r_{yb_k} n_{xb_k}] \sum_{i=1}^{M_{jk}} A_{b_i} \end{aligned} \quad (133)$$

The total pressure moment coefficients on the exit plane are

$$\begin{aligned} C_{mx2_k} &= \sum_{j=1}^{N_k} C_{mx2_{jk}} \\ C_{my2_k} &= \sum_{j=1}^{N_k} C_{my2_{jk}} \\ C_{mz2_k} &= \sum_{j=1}^{N_k} C_{mz2_{jk}} \end{aligned} \quad (134)$$

The momentum moment coefficients per column on the barrier are given by

$$C_{mx3_{jk}} = \frac{2}{b_r S_r} \sum_{i=1}^{M_{jk}} q_{b_i} [r_{y_i} v_{zb_i} - r_{z_i} v_{yb_i}] A_{b_i}$$

$$C_{my3_{jk}} = \frac{2}{c_r S_r} \sum_{i=1}^{M_{jk}} q_{b_i} [r_{z_i} v_{xb_i} - r_{x_i} v_{zb_i}] A_{b_i} \quad (135)$$

$$C_{mz3_{jk}} = \frac{2}{b_r S_r} \sum_{i=1}^{M_{jk}} q_{b_i} [r_{x_i} v_{yb_i} - r_{y_i} v_{xb_i}] A_{b_i}$$

The total momentum moment coefficients on the barrier are

$$C_{mx3_k} = \sum_{j=1}^{N_k} C_{mx3_{jk}}$$

$$C_{my3_k} = \sum_{j=1}^{N_k} C_{my3_{jk}} \quad (136)$$

$$C_{mz3_k} = \sum_{j=1}^{N_k} C_{mz3_{jk}}$$

The momentum moment coefficients per column on the exit plane are given by

$$C_{mx4_{jk}} = \frac{2}{b_r S_r E} \sum_{i=1}^{M_{jk}} q_{b_i}^2 [(r_{y_i} - n_{yb_k} h_k) t_{z_k} - (r_{z_i} - n_{zb_k} h_k) t_{y_k}] A_{b_i}$$

$$C_{my4_{jk}} = \frac{2}{c_r S_r E} \sum_{i=1}^{M_{jk}} q_{b_i}^2 \left[ (r_{z_i} - n_{zb_k} h_k) t_{x_k} - (r_{x_i} - n_{xb_k} h_k) t_{z_k} \right] A_{b_i} \quad (137)$$

$$C_{mz4_{jk}} = \frac{2}{b_r S_r E} \sum_{i=1}^{M_{jk}} q_{b_i}^2 \left[ (r_{x_i} - n_{xb_k} h_k) t_{y_k} - (r_{y_i} - n_{yb_k} h_k) t_{x_k} \right] A_{b_i}$$

The total momentum moment coefficients on the exit plane are

$$C_{mx4_k} = \sum_{j=1}^{N_k} C_{mx4_{jk}}$$

$$C_{my4_k} = \sum_{j=1}^{N_k} C_{my4_{jk}} \quad (138)$$

$$C_{mz4_k} = \sum_{j=1}^{N_k} C_{mz4_{jk}}$$

The pressure moment coefficients on the centerbody base are given by

$$C_{mxc_k} = \frac{1}{b_r S_r} C_{pc_k} \left[ r_{yb_k} n_{zb_k} - r_{zb_k} n_{yb_k} \right] A_{c_k}$$

$$C_{myc_k} = \frac{1}{c_r S_r} C_{pc_k} \left[ r_{zb_k} n_{xb_k} - r_{xb_k} n_{zb_k} \right] A_{c_k} \quad (139)$$

$$C_{mzc_k} = \frac{1}{b_r S_r} C_{pc_k} \left[ r_{xb_k} n_{yb_k} - r_{yb_k} n_{xb_k} \right] A_{c_k}$$

The lift, drag, and side force coefficients per column on the barrier and exit plane are given by

$$C_{L\ell_{jk}} = -C_{Fx\ell_{jk}} \sin \alpha \cos \psi + C_{Fy\ell_{jk}} \sin \alpha \sin \psi \\ + C_{Fz\ell_{jk}} \cos \alpha \quad (140)$$

$$C_{D\ell_{jk}} = C_{Fx\ell_{jk}} \cos \alpha \cos \psi - C_{Fy\ell_{jk}} \cos \alpha \sin \psi \\ + C_{Fz\ell_{jk}} \sin \alpha \quad (141)$$

$$C_{Y\ell_{jk}} = C_{Fx\ell_{jk}} \sin \psi + C_{Fy\ell_{jk}} \cos \psi \quad (142)$$

where  $\ell = 1$ , pressure force on barrier

$\ell = 2$ , pressure force on exit plane

$\ell = 3$ , momentum force on barrier

$\ell = 4$ , momentum force on exit plane

$\ell = c$ , pressure force on centerbody base

The total lift, drag, and side force coefficients are

$$C_{L\ell_k} = \sum_{j=1}^{N_k} C_{L\ell_{jk}} \quad (143)$$

$$C_{D\ell_k} = \sum_{j=1}^{N_k} C_{D\ell_{jk}} \quad (144)$$

$$C_{Y\ell_k} = \sum_{j=1}^{N_k} C_{Y\ell_{jk}} \quad (145)$$

where  $\ell = 1, 2, 3, 4$ , and  $c$ , as before

The wetted area per column of the barrier and exit plane is

$$A_{wb_{jk}} = \sum_{i=1}^{M_{jk}} A_{b_i} \quad (146)$$

$$A_{we_{jk}} = A_{wb_{jk}} \quad (147)$$

And the total wetted area of the barrier and exit plane is

$$A_{wb_k} = \sum_{j=1}^{N_k} A_{wb_{jk}} \quad (148)$$

$$A_{we_k} = A_{wb_k} \quad (149)$$

### 3. Total Force and Moment Coefficients of the Potential-Flow Model

The total force and moment coefficients of the model are composed of contributions due to all the source panels and all the lift fans.

$$C_{Fx} = C_{Fxs} + \sum_{k=1}^N [C_{Fx1_k} + C_{Fx2_k} + C_{Fx3_k} + C_{Fx4_k} + C_{Fxc_k}] \quad (150)$$

$$C_{Fy} = C_{Fys} + \sum_{k=1}^N [C_{Fy1_k} + C_{Fy2_k} + C_{Fy3_k} + C_{Fy4_k} + C_{Fyc_k}] \quad (151)$$

$$C_{Fz} = C_{Fzs} + \sum_{k=1}^N [C_{Fz1_k} + C_{Fz2_k} + C_{Fz3_k} + C_{Fz4_k} + C_{Fzc_k}] \quad (152)$$

$$C_{mx} = C_{mxs} + \sum_{k=1}^N [C_{mx1_k} + C_{mx2_k} + C_{mx3_k} + C_{mx4_k} + C_{mxc_k}] \quad (153)$$

$$C_{my} = C_{mys} + \sum_{k=1}^N [C_{my1_k} + C_{my2_k} + C_{my3_k} + C_{my4_k} + C_{myc_k}] \quad (154)$$

$$C_{mz} = C_{mzs} + \sum_{k=1}^N [C_{mz1_k} + C_{mz2_k} + C_{mz3_k} + C_{mz4_k} + C_{mzc_k}] \quad (155)$$

$$C_L = C_{Ls} + \sum_{k=1}^N [C_{L1_k} + C_{L2_k} + C_{L3_k} + C_{L4_k} + C_{Lc_k}] \quad (156)$$

$$C_D = C_{Ds} + \sum_{k=1}^N [C_{D1_k} + C_{D2_k} + C_{D3_k} + C_{D4_k} + C_{Dc_k}] \quad (157)$$

$$C_Y = C_{Ys} + \sum_{k=1}^N [C_{Y1_k} + C_{Y2_k} + C_{Y3_k} + C_{Y4_k} + C_{Yc_k}] \quad (158)$$

$$A_w = A_{ws} + \sum_{k=1}^N [A_{wb_k} + A_{we_k} + A_{c_k}] \quad (159)$$

where N = number of lift fans in the model

If the flow is symmetrical, only  $C_{Fx}$ ,  $C_{Fz}$ ,  $C_{my}$ ,  $C_L$ ,  $C_D$ , and  $A_w$  are calculated.

#### USAGE:

COMMON (See program MAIN for blank COMMON description.)

COMMON /JOM1/ XBP(500), YBP(500), ZBP(500),  
ANX(500), ANY(500), ANZ(500),  
AREA(500), VF(500), CP(500),  
SIGMA1(500)

COMMON /JOM2/ (See subroutine CONSUR.)

COMMON /JOM3/ XB(20), YB(20), ZB(20), VBX(20),  
VBY(20), VBZ(20), CPB(20), AR(20)

CALL FORCES (KSOL, XR, YR, ZR, SR, CR, BR)

Input: ICOM(1) = 1, symmetric flow  
               = 3, unsymmetric flow  
 ICOM(7) = number of lift fans in the model  
 NTSOUT = output data file number  
 NT16 = data file number of the quantities required to compute the force and moment coefficients  
 NETSP = number of source-panel networks  
 MSP } = arrays containing the dimensions of  
 NSP } the source-panel networks  
 ALPHA = array of angles of attack,  $\alpha$   
 PSI = array of angles of yaw,  $\psi$   
 XBP } = arrays of boundary-point coordinates  
 YBP } of the source-panel singularities  
 ZBP }  
 ANX } = arrays of unit normal vector components  
 ANY } of the source-panel singularities  
 ANZ }  
 VF = arrays of velocities at the source-panel boundary points  
 CP = arrays of pressure coefficients at the source-panel boundary points  
 SIGMA1 = arrays of singularity strengths of the source panels  
 TYPE = array of codes to indicate fan definition  
        = 0, center of fan in the x-z plane and flow is symmetric  
        = 1, center of fan not in the x-z plane and/or flow is unsymmetric  
 NR } = arrays containing the dimensions of  
 NTHETA } barrier area networks  
 CPC = array of assumed centerbody pressure coefficients

**CPE** = array of assumed exit plane pressure coefficients  
**XBC** }     = arrays of coordinates of the centers  
**YBC** }     of the lift-fan barriers  
**ZBC** }  
  
**ANBX** }     = arrays of unit normal vector com-  
**ANBY** }     ponents of the lift-fan barriers  
**ANBZ** }  
  
**ANEX** }     = arrays of unit vector components in  
**ANEY** }     the direction of the jet efflux  
**ANEZ** }  
  
**H**     = array of average distances between  
            the barriers and exit planes  
  
**DC**     = array of diameters of the centerbody  
            bases  
  
**XB** }     = arrays of boundary-point coordinates  
**YB** }     of the barrier areas  
**ZB** }  
  
**VBX** }     = arrays of velocity components at the  
**VBY** }     barrier boundary points  
**VBZ** }  
  
**CPB**     = arrays of pressure coefficients at the  
            barrier boundary points  
  
**AR**     = arrays of the barrier areas  
  
**KSOL**     = solution number  
  
**XR** }     = moment reference coordinates  
**YR** }  
**ZR** }  
  
**SR**     = reference planform area  
  
**CR**     = reference chord  
  
**BR**     = reference span

**SUBPROGRAMS CALLED:**     None  
**ERROR RETURNS:**     None  
**RESTRICTIONS:**     None

SUBJECT:

FORTRAN IV Subroutine GEOM

PURPOSE:

To control flow through the Geometric Section of the program

METHOD:

Control is accomplished through the use of control cards read and interpreted by function INTURP. Control cards used in the Geometric Section contain the following words in columns 1 through 4: SOUR, QUAD, MULT, OFF<sub>b</sub> (or OFF-) and END<sub>b</sub> where b represents a blank. When a control card is encountered, subroutine GEOM calls the appropriate subroutine (see Figure 71). For example, when a control card with the word SOUR in columns 1 through 4 is encountered, subroutine GEOM calls SPGEO, a subroutine that calculates the source-panel geometric properties.

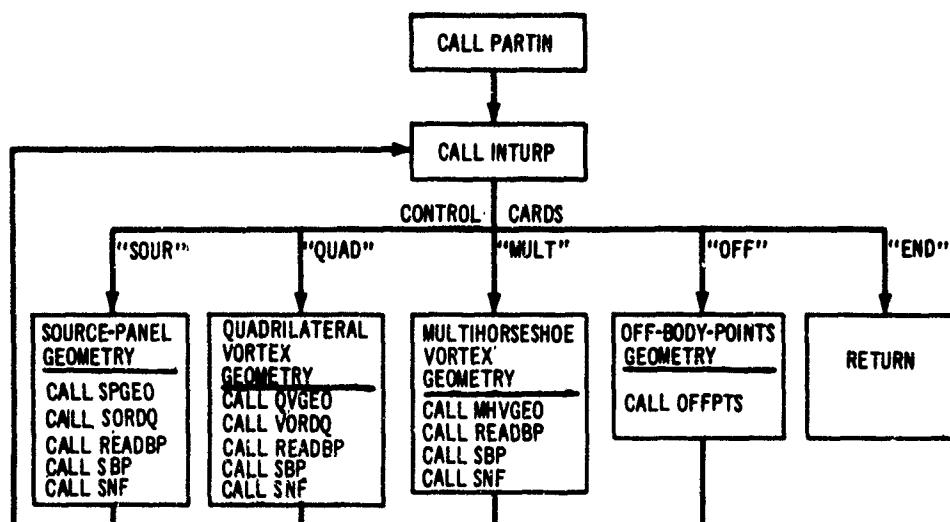


Figure 71. Subroutine GEOM— Flow Diagram.

USAGE:

COMMON (See program MAIN for blank COMMON description.)

CALL GEOM (IRROR)

Input: TITLE = any desired title

COM(1) } = date of run  
COM(2) }

ICOM(1) = 1, symmetric flow  
= 3, unsymmetric flow

NTSIN = input data file number  
NTSOUT = output data file number  
NT8 = data file number of the boundary-point quantities  
NT10 = data file number of the singularity-defining quantities  
**Output:** NROW = number of row elements in the velocity component matrices (number of singularities plus number of off-body points)  
NCOL = number of column elements in the velocity component matrices (number of singularities)  
IRROR = 0, successful  
= 1, geometry error detected

**SUBPROGRAMS CALLED:**

PARTIN  
INTURP  
SPGEO  
QVGEO  
MHVGE0  
OFFPTS

**ERROR RETURNS:** This routine transmits the Geometric Section error code to subroutine CTRL.

**RESTRICTIONS:** None

**SUBJECT:** FORTRAN IV Function INTURP

**PURPOSE:** To read and write a data card, then to find which four-character word in a given table is the same as the characters in columns 1 through 4 of the data card

**METHOD:** INTURP reads the next data card on the input data file with FORMAT (1A4). The contents of the card are written on the output data file with FORMAT (1H1, 1A4). A fixed-point comparison is made between the characters in columns 1 through 4 on the card and each four-character word in the table until a match is found or the end of the table is reached. The function value returned is the position in the table that matched the data card word, or is zero if no match was found.

**USAGE:** DIMENSION IDICT(N)

```

IGO = INTURP (IDICT, N, NTSIN, NTSOUT)

Input: IDICT      = table of four-character words
       N          = number of words in the table
       NTSIN     = input data file number
       NTSOUT    = output data file number

Output: IGO        = function value
```

**EXAMPLE:** A data deck might have three types of data cards, each group with a header card having a code word in the first four columns and perhaps comments in the remaining columns. Assume the code words are GEOM, AERO, and EXIT. A portion of the program could be written as follows:

```

DIMENSION IDICT(3)
DATA IDICT/4HGEOM, 4HAERO, 4HEXIT/
10 IGO = INTURP (IDICT, 3, NTSIN, NTSOUT)
   IF (IGO) 10, 10, 20
   20 GO TO (100, 200, 300), IGO
```

**SUBPROGRAMS CALLED:** None

**ERROR RETURNS:** None

**RESTRICTIONS:** None

**SUBJECT:** FORTRAN IV Program MAIN

**PURPOSE:** To initialize the program and to assign numbers to the data files

**METHOD:** The program performs the following operations:

1. Establishes blank COMMON, which is used throughout the program.
2. Obtains the data and stores it in COMMON.
3. Initializes certain program parameters, including specification of the maximum matrix partition size, and stores them in COMMON.
4. Assigns numbers to the data files (either magnetic tape units or disk files).
5. Calls subroutine CTRL, which controls the flow through the program.

Descriptions of blank COMMON and the data files follow.

#### BLANK COMMON DESCRIPTION

COMMON TITLE(8), COM(10), ICOM(10)  
COMMON NT1, NT2, NT3, NT4, NTSIN, NTSOUT, NT7  
COMMON NT8, NTPLT, NT10, NT11, NT12, NT13  
COMMON NT14, NT15, NT16, NT17, NT18, NT19, NT20  
COMMON NPR, NPC, LS(40), NROW, NCOL, MATSP  
COMMON MATQV, MATMHV, NETSP, MSP(200)  
COMMON NSP(200), NETQV, MQV(200), NQV(200)  
COMMON NETMHV, NHV(100), NRHS  
COMMON ALPHA(5), PSI(5), FSVX(5), FSVY(5), FSVZ(5)

Name	Explanation
TITLE	any desired title
COM(1) } COM(2) }	date of run
COM(3)	longest source-panel diagonal
COM(4) } to COM(10) }	not defined
ICOM(1)	{ =1, symmetric flow =3, unsymmetric flow
ICOM(2)	maximum matrix partition size

<u>Name</u>	<u>Explanation</u>
ICOM(3)	force and moment option code
ICOM(4)	plot option code
ICOM(5)	not defined
ICOM(6)	data case number
ICOM(7)	number of lift fans in the model
ICOM(8)	number of streamlines desired
ICOM(9)	matrix operating instruction data file indicator
ICOM(10)	not defined
NT1 NT2 NT3 NT4 NTSIN NTSOUT NT7 NT8 NTPLOT NT10 NT11 NT12 NT13 NT14 NT15 NT16 NT17 NT18 NT19 NT20	data file names
NPR	number of row partitions in the velocity component matrices
NPC	number of column partitions in the velocity component matrices
LS	array containing the number of row (column) elements in each row (column) partition

<u>Name</u>	<u>Explanation</u>
NROW	number of row elements in the velocity component matrices (number of singularities plus number of off-body points)
NCOL	number of column elements in the velocity component matrices (number of singularities)
MATSP	number of source-panel singularities
MATQV	number of quadrilateral vortex singularities
MATMHV	number of multihorseshoe vortex singularities
NETSP	number of source-panel networks
MSP } NSP }	arrays containing the dimensions of the source-panel networks
NETQV	number of quadrilateral vortex networks
MQV } NQV }	arrays containing the dimensions of the quadrilateral vortex networks
NETMHV	number of multihorseshoe vortex networks
NHV	array containing the dimensions of the multihorseshoe vortex networks
NRHS	number of simultaneous solutions desired (angle of attack/angle of yaw specification or zero free-stream velocity specification)
ALPHA	array of angles of attack, $\alpha$
PSI	array of angles of yaw, $\psi$
FSVX	array of x component of the free-stream velocities, $V_{\infty x}$
FSVY	array of y component of the free-stream velocities, $V_{\infty y}$
FSVZ	array of z component of the free-stream velocities, $V_{\infty z}$

#### DATA FILE DESCRIPTION

<u>Name</u>	<u>File Number</u>	<u>Contents of File</u>
NT1	1	singularity strengths, $\sigma$
NT2	2	x component of resultant-induced velocities, u
NT3	3	y component of resultant-induced velocities, v
NT4	4	z component of resultant-induced velocities, w
NTSIN	5	input data
NTSOUT	6	output data
NT7	7	instructions for matrix operations
NT8	8	boundary-point quantities
NTPLOT	9	data to be plotted
NT10	10	defining quantities of the singularities
NT11	11	velocity component matrix $[VX_{ij}]$
NT12	12	velocity component matrix $[VY_{ij}]$
NT13	13	velocity component matrix $[VZ_{ij}]$
NT14	14	punched output data for the boundary-layer program
NT15	15	specified weighting values for a network of multihorseshoe vortices
NT16	16	data for computation of the force and moment coefficients (In the Geometric Section of the program, this file contains the specified boundary-point quantities.)

<u>Name</u>	<u>File Number</u>	<u>Contents of File</u>
NT17	17	data for computation of the potential-flow properties of the lift fans (in the Geometric Section of the program, this file contains the specified normal velocities.)
NT18	18	
NT19	19	
NT20	20	additional multihorseshoe vortex-defining quantities

**USAGE:** Program MAIN is never referenced in a CALL statement.

**SUBPROGRAMS  
CALLED:** DATE  
CONTRL

**ERROR RETURNS:** None

**RESTRICTIONS:** None

**SUBJECT:** FORTRAN IV Subroutine MATDAT

**PURPOSE:** To store the matrix operating instructions in a data file

**METHOD:** The potential-flow program uses the Boeing-developed matrix routines (see subroutine MATRIX) to solve the normal velocity matrix equation for the unknown singularity strengths and to compute the resultant-induced velocities. Subroutine MATDAT supplies the matrix routines with instructions necessary to obtain the solution. The first call to MATDAT stores the instructions in a data file, rewinds the file, and changes an indicator (NTL01) from 0 to 1. Future calls will rewind the data file, but the indicator will prevent the instructions from being rewritten.

**USAGE:** CALL MATDAT (NTL01, NT7)

Input:      NTL01 = matrix operating instruction data file indicator

              NT7    = data file number of the matrix operating instructions

**SUBPROGRAMS CALLED:** None

**ERROR RETURNS:** None

**RESTRICTIONS:** None

**SUBJECT:** FORTRAN IV Subroutine MATRIX  
**PURPOSE:** To solve the normal velocity matrix equation for the unknown singularity strengths and to compute the resultant-induced velocities at the boundary points and off-body points  
**METHOD:** This routine calls the Boeing-developed matrix routines to do all the calculations.  
 The boundary condition of parallel flow to the surface at each boundary point is used to establish a matrix equation for the unknown strengths:

$$\begin{bmatrix} VN_{ij} \end{bmatrix} \begin{bmatrix} \sigma_{jk} \end{bmatrix} = \begin{bmatrix} B_{ik} \end{bmatrix} \quad (160)$$

where  $VN_{ij}$  = normal velocity induced at the  $i^{\text{th}}$  boundary point by the  $j^{\text{th}}$  unit strength singularity

$\sigma_{jk}$  = strength of the  $j^{\text{th}}$  singularity that satisfies the  $k^{\text{th}}$  solution

$B_{ik}$  = required normal velocity at the  $i^{\text{th}}$  boundary point for the  $k^{\text{th}}$  solution. It is the sum of the free-stream normal velocity component and the specified normal velocity

The resultant-induced velocities are given by

$$\begin{aligned} u_{ik} &= \sum_j V X_{ij} \cdot \sigma_{jk} \\ v_{ik} &= \sum_j V Y_{ij} \cdot \sigma_{jk} \\ w_{ik} &= \sum_j V Z_{ij} \cdot \sigma_{jk} \end{aligned} \quad (161)$$

where  $\left. \begin{array}{l} V X_{ij} \\ V Y_{ij} \\ V Z_{ij} \end{array} \right\}$  = velocity components induced at the  $i^{\text{th}}$  boundary point or off-body point by the  $j^{\text{th}}$  unit strength singularity

**USAGE:** CALL MATRIX (IRROR)

Output: IRROR = 0, if successful  
= 1, if an error occurred

SUBPROGRAMS  
CALLED:

Boeing-developed matrix routines

ERROR RETURNS: If an error occurred during execution of a particular matrix instruction, a message will be printed by one of the matrix subroutines. The format of the message appears below.

AN ERROR HAS OCCURRED IN TL-01 -  
THE CODE =  $n_1$

A TOTAL OF  $n_2$  INSTRUCTIONS WERE SUCCESS-  
FULLY EXECUTED PRIOR TO THE ERROR.

THE ABOVE COUNT WAS STARTED WITH THE  
CORE PROGRAM BELOW.

THE ERROR OCCURRED AT PHYSICAL  
INSTRUCTION NUMBER  $n_3$ .

A listing of the matrix instruction currently in use is given.

The final message is

THE ABOVE ERROR OCCURRED IN TL-01 WHILE  
TRYING TO EXECUTE THE FOLLOWING  
INSTRUCTION.

$n_4B \ n_5B \ n_6B \ n_7B \ n_8B$

In the above messages,  $n_1$  through  $n_8$  represent integers. The error code definitions for  $n_1$  are listed below.

<u>Code</u>	<u>Meaning</u>
1	The matrices are nonconformable.
2	One of the matrices is not square.
3	The matrix referenced by $n_4$ has illegal dimensions.
4	The matrix referenced by $n_5$ has illegal dimensions.
5	The matrix referenced by $n_4$ or $n_5$ is not diagonal.
6	A core program to be executed has illegal dimensions-- must be M x 6.
7	The program attempted to print a 0 x 0 matrix.
8	Overflow during a floating point multiply or add was attempted.
9	Overflow during a floating point divide was attempted.

<u>Code</u>	<u>Meaning</u>
10	An element of the matrix is 0 when attempting inversion.
11	The program attempted to invert a null matrix.
12	An element of the matrix is negative when trying to obtain square root of a diagonal.
13	Illegal data file number was found.
14	Legal data file number found, but it cannot be used for data storage.
15	Segmentation loader error was detected.
16	$n_5$ or $n_6$ is zero while trying to read instructions.
17	Binary tape error was detected—name check failed.
18	Binary tape error was detected—check sum error.
19	Storage limit of 32,563 words was exceeded.
20	Illegal number in $n_8$ of an instruction was detected.
21	The program attempted to partition below storage location 8000 or above 32,563.
22	A position number was referenced without having previously formed a partition.
23	The program attempted to store an instruction in an illegal location.
24	An illegal instruction was given during a data file manipulation.
25	Negative count in a cycle instruction was detected.
26	A cycle count greater than 1000 occurred.
27	Double precision storage trap (machine error) occurred.
28	Illegal dimensions were read from a data file.
29	The program referenced a storage location below 8000 or above 32,563.
30	Negative count of cells for clearing was detected.
31	Summing matrix is nonconformable with the product.
32	Inversion error was detected; the codes have been printed.
33	$n_4$ of a fixed-point increment/decrement instruction has a field count greater than 5.
34	The matrix has too many columns.
35	Illegal option number in an inversion instruction was detected.

<u>Code</u>	<u>Meaning</u>
100 through 105	An error occurred while reading one instruction at a time from a data file. If the error occurred on the first instruction, a 1,000,000 will appear in $n_g$ with all other numbers equal to 0.
1000 + i	Backspace files error where i is the number of files remaining when beginning of the file is reached.
2000 + i	Forward space files error where i is the number of unspaced files.
3000 + i	Forward space matrices error where i is the number of matrices remaining to be read.
1,000,000 through 1,002,000	Error code detected while loading program or reading from data files.
2,000,000 +1000 i + j	During the search for position i, illegal dimensions were encountered in position j (dimensions of 0 x 0, M x 0, or 0 x N will not cause this error).  If an error occurs, (1) check the data, (2) verify the matrix data files, and (3) rerun the case.

**RESTRICTIONS:** Maximum size of the normal velocity matrix (i.e., the number of singularities) should not exceed 1200.

**SUBJECT:** FORTRAN IV Subroutine MHVGEO

**PURPOSE:** To calculate the multihorseshoe vortex geometric quantities required by the Aerodynamic Section of the program and to store them in partition form in data files

**METHOD:**

The geometric quantities are separated into two arrays: the singularity-defining quantities array and the boundary-point quantities array. Each multihorseshoe vortex has a singularity-defining quantities array and a boundary-point quantities array. The two arrays are stored in separate data files for use by the Aerodynamic Section. The defining quantities of a multihorseshoe vortex may easily exceed the 40 allowed in the singularity-defining quantities array; consequently, additional multihorseshoe vortex-defining quantities arrays are established. These additional arrays are stored in a maximum of three additional multihorseshoe vortex-defining quantities data files.

The singularity-defining quantities data file (the one with 40 quantities) contains multihorseshoe vortex-defining quantities and defining quantities of the other singularities (source panels and quadrilateral vortices). The source-panel-defining quantities arrays are stored first, followed by the quadrilateral vortex arrays and the multihorseshoe vortex arrays. Each array (the defining quantities for one singularity) is used by the Aerodynamic Section to calculate one column of the aerodynamic velocity component matrices. Because of limited core storage, the quantities are stored as a number of partitioned arrays; this storage permits the computation of the aerodynamic velocity components as partitions of the entire matrix.

The number of defining quantities arrays in each partition is determined from the COMMON array LS, which contains the number of column elements in each column partition (see subroutine PARTIN). The numbers in the array are used sequentially until the defining quantities of all the singularities are stored in the singularity-defining quantities data file.

The boundary-point quantities data file is formed in the same manner, except that the words "row" and "column" are interchanged and the off-body point quantities, if any, are added to the data file immediately following the multihorseshoe vortex boundary-point quantities.

The additional multihorseshoe vortex-defining quantities data files are formed in a special manner described later.

Subroutine MHVGEO is concerned only with storing the multihorseshoe vortex geometric quantities in the data files. MHVGEO begins by reading all the multihorseshoe vortex corner points of a network from the input data file. Then the array(s) of weighting values is (are) read and stored in a data file. Subroutine READBP, which reads in the specified boundary points, is called and, if requested, reads the specified normal velocities of the network vortices and stores them in separate data files.

Now, MHVGEO treats each multihorseshoe vortex individually. First, the network indexes associated with the vortex are replaced by partition index numbers (this will help in storing the geometric quantities in partition form in the data files). Then the singularity-defining quantities, which consist of parameters such as the number of corner points and the number of weighted vortex segments, are calculated. There is also space in the defining quantities array for 31 weighting values. The coordinates of the corner points and any excess weighting values, if the number of weighted vortex segments exceeds 32, are stored in an additional multihorseshoe vortex-defining quantities data file.

A call to subroutine SBP is made, which places the specified boundary-point quantities in the boundary-point quantities array. If the option to specify the normal velocities is exercised, a call to subroutine SNF is made, which stores the specified normal velocities of the vortex in the boundary-point quantities array. Finally, the singularity-defining quantities, the additional multihorseshoe vortex-defining quantities, and the boundary-point quantities are printed to aid in the detection of input errors. This procedure is repeated for every multihorseshoe vortex of the potential-flow model.

As mentioned previously, the singularity-defining quantities (not the additional multihorseshoe vortex-defining quantities) and the boundary-point quantities are stored in the core until the end of a partition is reached, which is determined by the partition index numbers, then the quantities are added to the data files. Quantities for the next partition are calculated and stored in the core and also added to the data files. The additional multihorseshoe vortex-defining quantities are stored in a data file as they are calculated. There is an additional multihorseshoe vortex-defining quantities data file for each partition. Since only three data files are assigned to the additional multihorseshoe vortex-defining quantities, care must be taken to ensure that no more than three partitions contain multihorseshoe singularities.

The singularity-defining quantities array and boundary-point quantities array are stored in the labeled COMMON block COM2. The corner-point coordinates for the singularities of a network are stored in the labeled COMMON block COM3.

USAGE:

COMMON (See program MAIN for blank COMMON description.)

COMMON/COM2/DQ(40, 50), BP(15, 50)

CALL MHVGEO (JPC, JC2, JCOL, IRROR)

Input: NTSIN = input data file number  
NTSOUT = output data file number  
NT8 = data file number of the boundary-point quantities  
NT10 = data file number of the singularity-defining quantities  
NT15 = data file number of the weighting values of a network  
NT16 = data file number of the specified boundary-point quantities of a network  
NT17 = data file number of the specified normal velocities of a network  
NT18 | NT19 | NT20 | = data file numbers of the additional multihorseshoe vortex-defining quantities  
NPC = number of row (column)\* partitions in the aerodynamic velocity component matrices  
LS = array containing the number of row (column)\* elements in each row (column)\* partition

"Row" refers to the boundary-point quantities, and "column" refers to the singularity-defining quantities.

**DQ**      = defining quantities for the last partition containing quadrilateral vortices. This array is used when the partition is only partly completed by the quadrilateral vortices  
**BP**      = boundary-point quantities for the last partition containing quadrilateral vortices. This array is used when the partition is only partly completed by the quadrilateral vortices  
  
**Input/**  
**Output:** **JPC**      = index number of the row (column)\* partition  
**JC2**      = index number of the row (column)\* element in the row (column)\* partition  
**JCOL**      = index number of the row (column)\* element in the aerodynamic velocity component matrices  
  
**Output:** **MATMHV** = total number of multihorseshoe singularities used in the potential-flow model  
**NHV**      = array containing the dimensions of the multihorseshoe vortex networks  
**IRROR**      = 0, routine successful  
                = 1, input error in the multihorseshoe vortex geometry

**SUBPROGRAMS  
CALLED:**  
**READBP**  
**SBP**  
**SNF**

**ERROR RETURNS:** If an error occurs, this routine returns to subroutine **CONTRL**, where an error message is written and execution is passed to the next case.

**RESTRICTIONS:** No more than three partitions can contain multihorseshoe singularities, since only three data files are assigned to store the additional multihorseshoe vortex-defining quantities.

**SUBJECT:** FORTRAN IV Subroutine OFFPTS

**PURPOSE:** To store the off-body point coordinates in the boundary-point quantities data file

**METHOD:** An off-body point is a specified point in the reference coordinate system where the flow properties are computed. Since off-body points are used to compute a portion of the velocity component matrices in the Aerodynamic Section of the program (see subroutines SFLOW and UFLOW), they must reside in partition form in the data file. This is accomplished by subdividing the off-body points into partitions of 50 points each, since 50 is the maximum matrix partition size. For example, an array of 257 off-body points is subdivided into six partitions—five partitions containing 50 points and one partition containing the remaining seven points.

The number of off-body point partitions is added to the number of row partitions previously calculated by subroutine PARTIN to give the total number of row partitions in the velocity component matrices. The array of numbers of points in the off-body point partitions is stored in the blank COMMON array LS.

Once the number of off-body point partitions and the number of points in each partition have been determined, OFFPTS reads all the off-body points from the input data file. The first partition of points is transferred to the boundary-point array which, in turn, is stored in the boundary-point quantities data file. This process continues until all of the off-body point partitions are stored in the data file.

This routine requires very little additional core storage since it uses previously established COMMON arrays to store all data. Initially, the off-body point coordinates are stored in the labeled COMMON block COM3. Then the points are transferred, one partition at a time, to the boundary-point quantities data file.

**USAGE:** COMMON (See program MAIN for blank COMMON description.)

CALL OFFPTS (MATOFF)

Input: ICOM(2) = maximum matrix partition size  
NTSIN = input data file number

**NTSOUT** = output data file number

**NT8** = data file number of the boundary-point quantities

**Input/Output:** **NPR** = number of row partitions in the velocity component matrices

**Output:** **LS** = array of numbers of points in the off-body partitions. This array is stored immediately below the array containing the number of elements in each row partition (previously calculated by subroutine PARTIN)

**MATOFF** = number of off-body points

**SUBPROGRAMS**

**CALLED:** None

**ERROR RETURNS:** None

**RESTRICTIONS:** None

**SUBJECT:**

**FORTRAN IV Subroutine PANEL**

**PURPOSE:**

To calculate the streamline characteristics of a specified source panel

**METHOD:**

The panel is identified by three indices,  $i$ ,  $j$ , and  $k$ , which locate the panel as the  $i^{th}$  position of the  $j^{th}$  column of the  $k^{th}$  network. Once the panel is identified, this routine determines its streamline entrance and exit sides. If the panel is on the periphery of a network, the peripheral side(s) is (are) given as a special identification number(s). Before determining the entrance and exit sides, the panel's velocity components in the element coordinate system are calculated using the transformation

$$\begin{aligned} v_\xi &= a_{11} v_x + a_{12} v_y + a_{13} v_z \\ v_\eta &= a_{12} v_x + a_{22} v_y + a_{23} v_z \end{aligned} \quad (162)$$

where  $\left. \begin{matrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{matrix} \right\} = \begin{matrix} \text{direction cosines of the element} \\ \text{coordinate system} \end{matrix}$

$\left. \begin{matrix} v_x \\ v_y \\ v_z \end{matrix} \right\} = \begin{matrix} \text{the velocity components in the} \\ \text{reference coordinate system} \end{matrix}$

A panel side is an entrance side if:

$$|\nabla \cdot \vec{d}|_e < 0 \quad (163)$$

where  $|\nabla \cdot \vec{d}|_e = v_\xi \cdot (\eta_{e+1} - \eta_e) - v_\eta \cdot (\xi_{e+1} - \xi_e)$ ,  
 $e = 1, 2, 3, 4$

If  $|\nabla \cdot \vec{d}|_e \geq 0$ , the side is an exit side.

If the panel is on the periphery of a network, a special code (IPERF<sub>l</sub>) identifies the peripheral side(s).

Initially,

$$IPERF_l = 0, l = 1, 2, 3, 4 \quad (164)$$

If  $j$  equals 1, side 1 is a peripheral edge (see Figure 72) and

$$IPERF_1 = 1 \quad (165)$$

If  $i$  equals  $M$ , side 2 is a peripheral edge and

$$IPERF_2 = 2 \quad (166)$$

If  $j$  equals  $N$ , side 3 is a peripheral edge and

$$IPERF_3 = 3 \quad (167)$$

If  $i$  equals 1, side 4 is a peripheral edge and

$$IPERF_4 = 4 \quad (168)$$

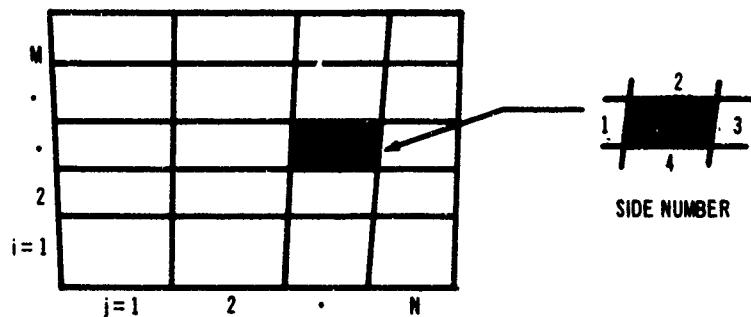


Figure 72. Subroutine PANEL—Source-Panel Network.

USAGE:

COMMON (See program MAIN for blank COMMON description.)

COMMON /SL1/ A11(1000), A12(1000), A13(1000),  
A21(1000), A22(1000), A23(1000), ANX(1000),  
ANY(1000), ANZ(1000), G1(1000), G2(1000), G3(1000),  
G4(1000), H1(1000), H2(1000), H3(1000), H4(1000),  
XBP(1000), YBP(1000), ZBP(1000), LPERF(1000),  
VFX(1000), VFY(1000), VFZ(1000)  
COMMON /SL2/ G(5), H(5), VG, VH, IVD(4), IPERF(4),  
VDEP, COSTP, E10, E11, E13

CALL PANEL (I, J, K, L)

Input:  $\{ \begin{matrix} \text{MSP} \\ \text{NSP} \end{matrix} \}$  = arrays containing dimensions of the source-panel networks  
 $\{ \begin{matrix} \text{A11} \\ \text{A12} \\ \text{A13} \\ \text{A21} \\ \text{A22} \\ \text{A23} \end{matrix} \}$  = arrays of direction cosines of the element coordinate systems  
 $\{ \begin{matrix} \text{G1} \\ \text{G2} \\ \text{G3} \\ \text{G4} \end{matrix} \}$  = arrays of  $\xi$  coordinates of the source-panel corner points  
 $\{ \begin{matrix} \text{H1} \\ \text{H2} \\ \text{H3} \\ \text{H4} \end{matrix} \}$  = arrays of  $\eta$  coordinates of the source-panel corner points  
**LPERF** = array of codes to indicate panels with sides on the network periphery  
                   = 0, panel not on network periphery  
                   = 1, panel on network periphery  
 $\{ \begin{matrix} \text{VFX} \\ \text{VFY} \\ \text{VFZ} \end{matrix} \}$  = arrays of the velocity components of the source panels in the reference coordinate system  
 I = panel position index  
 J = panel column index  
 K = panel network index  
 Output: G = array of  $\xi$  coordinates of the specified source panel  
 H = array of  $\eta$  coordinates of the specified source panel  
 $\{ \begin{matrix} \text{VG} \\ \text{VH} \end{matrix} \}$  = velocity components of the specified source panel in the element coordinate system  
 IVD = array identifying the entrance and exit sides  
                   = 0, entrance side  
                   = 1, exit side

**IPERF** = array identifying the peripheral  
sides of the specified panel  
= 0, not a peripheral side  
= *i*, a peripheral side, where *i* is  
the side number

**L** = panel accumulative number

**SUBPROGRAMS**  
**CALLED:** None

**ERROR RETURNS:** None

**RESTRICTIONS:** None

**SUBJECT**                   **FORTRAN IV Subroutine PARTIN**

**PURPOSE:**               To partition a square matrix

**METHOD:**               Given the size of a square matrix and the maximum partition size, this routine optimally partitions the matrix. Three constraints are used: (1) the number of partitions must be kept as small as possible, (2) the partitions must be nearly equal in size, and (3) no partition can exceed the maximum size. For example, a typical fan-in-wing potential-flow model requiring 1093 singularities results in the formation of a 1093 by 1093 aerodynamics velocity matrix, which must be partitioned into submatrices no larger than 50 by 50 because of core limitations. PARTIN, within the constraints listed above, partitions the matrix into 484 partitions (submatrices). The 484 partitions form a matrix of 22 row partitions by 22 column partitions. The number of row elements in each row partition is determined and stored in an array. This array applies equally to the column partitions, since they are identical to the row partitions.

**USAGE:**               COMMON (See program MAIN for blank COMMON description.)

**CALL PARTIN (NSING)**

<b>Input:</b>	ICOM(2)     = maximum matrix partition size
	NTSOUT     = output data file number
	NSING       = matrix size (also equals the total number of singularities used)
<b>Output:</b>	NPR         = number of row partitions
	NPC         = number of column partitions (Note: for a square matrix, NPR = NPC.)
	LS           = array containing the number of row (column) elements in each row (column) partition

**SUBPROGRAMS CALLED:**       None

**ERROR RETURNS:**           None

**RESTRICTIONS:**           Maximum number of row (column) partitions is 40.

**SUBJECT:** FORTRAN IV Subroutine PLOT

**PURPOSE:** To plot the pressure distributions on the body

**METHOD:** This is not an operational routine since plotting depends on the available plotting facilities and software at the user's installation. The quantities to be plotted, however, are stored on a data file where they may easily be retrieved by a plot routine tailored to the user's installation. A partial listing of the plotting routine used at the Commercial Airplane Division of The Boeing Company is provided.

**USAGE:** CALL PLOT

**SUBPROGRAMS  
CALLED:** None

**ERROR RETURNS:** None

**RESTRICTIONS:** None

**SUBJECT:** FORTRAN IV Subroutine QVGEO

**PURPOSE:** To calculate the quadrilateral vortex geometric quantities required by the Aerodynamic Section of the program and to store them in partition form in data files

**METHOD:**

The geometric quantities are separated into two arrays: the singularity-defining quantities array and the boundary-point quantities array. Each quadrilateral vortex has a singularity-defining quantities array and a boundary-point quantities array. The two arrays are stored in separate data files for use by the Aerodynamic Section. The singularity-defining quantities data file contains the quadrilateral vortex-defining quantities and the defining quantities of the other singularities (source panels and multi-horseshoe vortices). The source-panel-defining quantities arrays are stored first, followed by the quadrilateral vortex arrays and the multihorseshoe vortex arrays. Each array (the defining quantities for one singularity) is used by the Aerodynamic Section to calculate one column of the aerodynamic velocity component matrices. Because of limited core storage, the quantities are stored as a number of partitioned arrays. This permits the computation of the aerodynamic velocity components as partitions of the entire matrix.

The number of defining quantities arrays in each partition is determined from the COMMON array LS, which contains the number of column elements in each column partition (see subroutine PARTIN). The numbers in the array are used sequentially until the defining quantities of all the singularities are stored on the singularity-defining quantities data file.

The boundary-point quantities data file is formed in the same manner, except that the words "row" and "column" are interchanged and the off-body point quantities, if any, are added to the data file immediately following the multi-horseshoe vortex boundary-point quantities.

Subroutine QVGEO is concerned only with storing the quadrilateral vortex geometric quantities in the data files. QVGEO begins by reading all the quadrilateral vortex corner points of a network from the input data file. Subroutine READBP is called and, if requested, reads in the specified boundary points and/or the specified normal velocities of the network vortices and stores them in separate data files.

At this point, QVGEO treats each vortex individually. First, the network indexes associated with the vortex are replaced by partition index numbers (this will help in storing the geometric quantities in partition form in the

SUBJECT: FORTRAN IV Subroutine QVGEO

PURPOSE: To calculate the quadrilateral vortex geometric quantities required by the Aerodynamic Section of the program and to store them in partition form in data files

METHOD: The geometric quantities are separated into two arrays: the singularity-defining quantities array and the boundary-point quantities array. Each quadrilateral vortex has a singularity-defining quantities array and a boundary-point quantities array. The two arrays are stored in separate data files for use by the Aerodynamic Section. The singularity-defining quantities data file contains the quadrilateral vortex-defining quantities and the defining quantities of the other singularities (source panels and multi-horseshoe vortices). The source-panel-defining quantities arrays are stored first, followed by the quadrilateral vortex arrays and the multihorseshoe vortex arrays. Each array (the defining quantities for one singularity) is used by the Aerodynamic Section to calculate one column of the aerodynamic velocity component matrices. Because of limited core storage, the quantities are stored as a number of partitioned arrays. This permits the computation of the aerodynamic velocity components as partitions of the entire matrix.

The number of defining quantities arrays in each partition is determined from the COMMON array LS, which contains the number of column elements in each column partition (see subroutine PARTIN). The numbers in the array are used sequentially until the defining quantities of all the singularities are stored on the singularity-defining quantities data file.

The boundary-point quantities data file is formed in the same manner, except that the words "row" and "column" are interchanged and the off-body point quantities, if any, are added to the data file immediately following the multi-horseshoe vortex boundary-point quantities.

Subroutine QVGEO is concerned only with storing the quadrilateral vortex geometric quantities in the data files. QVGEO begins by reading all the quadrilateral vortex corner points of a network from the input data file. Subroutine READBP is called and, if requested, reads in the specified boundary points and/or the specified normal velocities of the network vortices and stores them in separate data files.

At this point, QVGEO treats each vortex individually. First, the network indexes associated with the vortex are replaced by partition index numbers (this will help in storing the geometric quantities in partition form in the

data files). Then QVGEO calls subroutine VORDQ, which calculates the defining quantities and the boundary-point quantities of the vortex. If the option to specify the boundary-point quantities is exercised, a call to subroutine SBP is made, and the specified boundary-point quantities are placed in the boundary-point quantities array. If the option to specify the normal velocities is exercised, a call to subroutine SNF is made, and the specified normal velocities of the vortex are placed in the boundary-point quantities array. Finally, selected geometric properties of the vortex are output to aid in the detection of input errors. This procedure is repeated for every quadrilateral vortex of the potential-flow model.

As mentioned previously, the defining quantities and the boundary-point quantities of the quadrilateral vortices are stored in the core until the end of a partition is reached (this is determined by the partition index numbers). When this occurs, the quantities are added to the data files. The geometric quantities for the next partition are calculated, stored in the core, and finally added to the data files. If the last quadrilateral vortex does not complete a partition (and it probably will not), the geometric quantities of the partition remain in the core until additional geometric quantities required for the multihorse-shoe vortex singularities (calculated by subroutine MHVGEO) complete the partition. Then the geometric quantities of the two types of singularities are added to the data files, where they are stored in partition form, uninterrupted by the change in singularities.

The singularity-defining quantities array and boundary-point array are stored in the labeled COMMON block COM2. The corner-point coordinates for the singularities of a network are stored in the labeled COMMON block COM3.

**USAGE:**

COMMON (See program MAIN for blank COMMON description.)

COMMON/COM2/DQ (40, 50), BP(15, 50)

CALL QVGEO (JPC, JC2, JCOL, IRROR)

Input: NTSIN = input data file number

NTSOUT = output data file number

NT8 = data file number of the boundary-point quantities

	<b>NT10</b>	= data file number of the singularity-defining quantities
	<b>NT16</b>	= data file number of the specified boundary-point quantities of a network
	<b>NT17</b>	= data file number of the specified normal velocities of a network
	<b>NPC</b>	= number of row (column)* partitions in the aerodynamic velocity component matrices
	<b>LS</b>	= array containing the number of row (column)* elements in each row (column)* partition
	<b>DQ</b>	= defining quantities for the last partition containing source panels. This is used when the partition is only partly completed by the source panels
	<b>BP</b>	= boundary-point quantities for the last partition containing source panels. This is used when the partition is only partly completed by the source panels
<b>Input/Output:</b>	<b>JPC</b>	= index number of the row (column)* partition
	<b>JC2</b>	= index number of the row (column)* element in the row (column)* partition
	<b>JCOL</b>	= index number of the row (column)* element in the aerodynamic velocity component matrices
<b>Output:</b>	<b>MATQV</b>	= total number of quadrilateral vortex singularities used in the potential-flow model
	<b>MQV   NQV  </b>	= arrays containing the dimensions of the quadrilateral vortex networks

**"Row"** refers to the boundary-point quantities, and **"column"** refers to the singularity-defining quantities.

DQ = defining quantities for the last partition containing quadrilateral vortices  
BP = boundary-point quantities for the last partition containing quadrilateral vortices  
IRROR = 0, routine successful  
= 1, input error in quadrilateral vortex geometry

SUBPROGRAMS  
CALLED:  
READBP  
VORDQ  
SBP  
SNF

ERROR RETURNS: If an error occurs, this routine returns to subroutine CTRL, where an error message is written and execution is passed to the next case.

RESTRICTIONS: None

**SUBJECT:** FORTRAN IV Subroutine READBP

**PURPOSE:** To read the specified boundary-point quantities and/or the specified normal velocities of a network of singularities and to store them in data files

**METHOD:** This subroutine is used by subroutines SPGEO, QVGEO, and MHVGEO to transfer data from the input data file to two data files. READBP tests the specified boundary-point option code OPTBP and the specified normal velocity option code OPTUS to determine how much data, if any, are to be transferred.

**USAGE:** CALL READBP (OPTBP, OPTUS, NSING, NTSIN, NT16, NT17)

Input: OPTBP = specified boundary-point option code  
OPTUS = specified normal velocity option code  
NSING = number of singularities in the network  
NTSIN = input data file number  
NT16 = data file number of the specified boundary-point quantities  
NT17 = data file number of the specified normal velocities

**SUBPROGRAMS CALLED:** None

**ERROR RETURNS:** None

**RESTRICTIONS:** None

**SUBJECT:** ASCENT Subroutine READTP  
**PURPOSE:** To read a matrix from a data file  
**METHOD:** The reading is done with a format consistent with Boeing-developed matrix routines.  
**USAGE:** DIMENSION A(JR, N), B(12)  
 CALL READTP (A (1, 1), JR, NAME, M, N, B, NF, NMT, NTAPE, IRROR)  
 Input: JR = maximum matrix size. This also must be the row DIMENSION statement entry for A.  
 M = number of rows in the matrix  
 N = number of columns in the matrix  
 B = scratch array for use by READTP  
 NF = number of end-of-file marks to be passed before reading starts. If 0, no file spacing takes place.  
 NMT = number of matrices to be passed before reading occurs. If 0, no matrix spacing takes place.  
 NTAPE = number of the data file containing the matrix  
 Output: A = the matrix to be read from the data file. If this is not A (1, 1), the designated matrix will be read from the data file.  
 NAME = integer name of the matrix  
 IRROR = error code  
**SUBPROGRAMS CALLED:** None  
**ERROR RETURNS:** IRROR = 0, if successful read  
                   = 1, if file spacing error  
                   = 2, if matrix spacing is negative  
                   = 3, if matrix spacing error occurs  
                   = 4, if check sum error  
                   = 5, if number of the data file containing the matrix is wrong  
**RESTRICTIONS:** None

SUBJECT:

FORTRAN IV Subroutine RHS

PURPOSE:

To calculate the elements of a partition of the required normal velocity matrix  $[B_{ik}]$

METHOD:

The required normal velocity  $B_{ik}$  is the sum of the free-stream normal velocity component and the specified normal velocity at the  $i^{\text{th}}$  boundary point for the  $k^{\text{th}}$  solution. The specified normal velocity is zero unless specifically changed by the program user. An element of the partition is given by

$$B_{ik} = (n_{x_i} V_{\infty x_k} + n_{y_i} V_{\infty y_k} + n_{z_i} V_{\infty z_k}) + U_{s_{ik}} \quad (169)$$

where  $n_{x_i}$   
 $n_{y_i}$   
 $n_{z_i}$  } = unit normal vector components at the  $i^{\text{th}}$  boundary point

$V_{\infty x_k}$   
 $V_{\infty y_k}$   
 $V_{\infty z_k}$  } = free-stream velocity components for the  $k^{\text{th}}$  solution

$U_{s_{ik}}$  = specified normal velocity at the  $i^{\text{th}}$  boundary point for the  $k^{\text{th}}$  solution

Subroutine RHS stores all the elements of the partition in the array VX, which appears in the labeled COMMON block KOM3. The unit normal vector components and the specified normal velocities are stored in the boundary-point quantities array BP, which appears in the labeled COMMON block KOM2.

**USAGE:** COMMON (See program MAIN for blank COMMON description.)

COMMON /KOM2/ DQ(40, 50), BP(15, 50)

COMMON /KOM3/ VX(50, 50), VY(50, 50), VZ(50, 50)

CALL RHS (NR2)

Input: NRHS = number of simultaneous solutions desired

FSVX } = arrays of the free-stream  
FSVY } = velocity components,  
FSVZ }  $(V_{\infty x_k}, V_{\infty y_k}, V_{\infty z_k})$

NR2 = number of row elements in the partition

BP = array of boundary-point quantities

Output: VX = required normal velocity matrix partition

**SUBPROGRAMS CALLED:** None

**ERROR RETURNS:** None

**RESTRICTIONS:** None

**SUBJECT:** FORTRAN IV Subroutine SBP

**PURPOSE:** To store and print the specified boundary-point quantities of a singularity

**METHOD:** The specified boundary-point coordinates ( $x, y, z$ ) and/or unit normal vector components ( $n_x, n_y, n_z$ ) are read from a data file (prepared by subroutine READBP) and stored in the singularity boundary-point quantities array. All specified quantities are defined in the reference coordinate system. The specified boundary-point quantities override the boundary-point quantities previously calculated by subroutine SORDQ or VORDQ. Except where overridden, the boundary-point quantities array remains unchanged. A list of the boundary-point quantities array is given below.

BP(1,JC2)	= x	specified boundary-point coordinates
BP(2,JC2)	= y	
BP(3,JC2)	= z	
BP(4,JC2)	= $n_x$	specified unit normal vector components at the boundary point
BP(5,JC2)	= $n_y$	
BP(6,JC2)	= $n_z$	
BP(7,JC2)	to BP(15,JC2)	remain unchanged

where JC2 denotes the row element number of a partition.

Finally, the specified boundary-point quantities are printed via the output data file. The boundary-point quantities array BP is passed to higher-level subroutines through a labeled COMMON statement.

**USAGE:**  
 COMMON /COM2/ DQ(40,50), BP(15,50)  
 CALL SBP (OPTBP, JC2, NT16, NTSOUT)

Input: OPTBP = specified boundary-point option code  
JC2 = row element number of a partition  
NT16 = data file number of the specified  
boundary-point quantities  
NTSOUT = output data file number  
Output: BP = boundary-point quantities array  
(only locations overridden by the  
specified quantities are output)

SUBPROGRAMS  
CALLED: None

ERROR RETURNS: None

RESTRICTIONS: None

**SUBJECT:****FORTRAN IV Subroutine SEARCH****PURPOSE:**

To search the periphery of the source-panel networks for a panel with an acceptable entrance side for the streamline

**METHOD:**

The selection of a new entrance panel involves a series of tolerance tests. If an acceptable panel is not found, the streamline is terminated.

Before conducting the tests, the exit point of the previous panel (last panel the streamline successfully crossed) is transformed to the reference coordinate system by

$$\begin{aligned}x_h &= x_p + a_{11} p \xi_{px} + a_{21} p \eta_{px} \\y_h &= y_p + a_{12} p \xi_{px} + a_{22} p \eta_{px} \\z_h &= z_p + a_{13} p \xi_{px} + a_{23} p \eta_{px}\end{aligned}\quad (170)$$

where

$$\left. \begin{array}{l} x_p \\ y_p \\ z_p \end{array} \right\} = \begin{array}{l} \text{boundary-point coordinates of the} \\ \text{previous panel in the reference} \\ \text{coordinate system} \end{array}$$

$$\left. \begin{array}{l} a_{11} p \quad a_{21} p \\ a_{12} p \quad a_{22} p \\ a_{13} p \quad a_{23} p \end{array} \right\} = \begin{array}{l} \text{direction cosines of the previous} \\ \text{panel in the element coordinate} \\ \text{system} \end{array}$$

$$\left. \begin{array}{l} \xi_{px} \\ \eta_{px} \end{array} \right\} = \begin{array}{l} \text{streamline exit point coordinates of} \\ \text{the previous panel in the element} \\ \text{coordinate system} \end{array}$$

The series of tolerance tests begin by comparing the distance between the previous panel exit point and the peripheral panel boundary point. Failure to pass any test eliminates the panel.

### Test 1

$$|x - x_h| < \epsilon_{13} \quad (171)$$

where  $\epsilon_{13}$  equals  $0.8t$  ( $t$  is the longest source-panel diagonal).

### Test 2

$$|y - y_h| < \epsilon_{13} \quad (172)$$

After passing these tests, the approximate streamline entrance point of the peripheral panel in the element coordinate system is computed as

$$\begin{aligned}\xi_h &= a_{11}(x_h - x) + a_{12}(y_h - y) + a_{13}(z_h - z) \\ \eta_h &= a_{21}(x_h - x) + a_{22}(y_h - y) + a_{23}(z_h - z) \\ \zeta_h &= a_{31}(x_h - x) + a_{32}(y_h - y) + a_{33}(z_h - z)\end{aligned}\quad (173)$$

where  $\left. \begin{matrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{matrix} \right\}$  = direction cosines of the peripheral panel in the element coordinate system

$\left. \begin{matrix} x \\ y \\ z \end{matrix} \right\}$  = boundary-point coordinates of the peripheral panel in the reference coordinate system

Now the sides on the periphery of a network are tested. The particular test depends on the panel's three indices,  $i$ ,  $j$ , and  $k$ , which locate the panel as the  $i^{\text{th}}$  position of the  $j^{\text{th}}$  column of the  $k^{\text{th}}$  network.

### Test 3

This test is conducted only if  $j = 1$ ; i.e., side one is on the periphery.

$$d_a + d_b - d_e < 0.01 d_e \quad (174)$$

and  $d_a/d_e \leq 1$  and  $d_b/d_e \leq 1$

$$\text{where } d_a = \sqrt{(t_h - \xi_1)^2 + (\eta_h - \eta_1)^2 + \xi_h^2}$$

$$d_b = \sqrt{(\xi_h - \xi_2)^2 + (\eta_h - \eta_2)^2 + \xi_h^2}$$

$$d_e = \sqrt{(\xi_1 - \xi_2)^2 + (\eta_1 - \eta_2)^2}$$

If this test is successful, the entrance side number and the fractional distance of the streamline starting point along the side are established ( $e = 1$  and  $d/d_e = d_a/d_e$ ), and the subroutine proceeds to Test 7.

#### Test 4

This test is conducted only if  $i = M$ ; i. e., side two is on the periphery. The test is identical to Test 3 except that  $\xi_1$  is replaced by  $\xi_2$ ,  $\xi_2$  by  $\xi_3$ ,  $\eta_1$  by  $\eta_2$ , and  $\eta_2$  by  $\eta_3$ .

#### Test 5

This test is conducted only if  $j = N$ ; i. e., side three is on the periphery. The test is identical to Test 3 except that  $\xi_1$  is replaced by  $\xi_3$ ,  $\xi_2$  by  $\xi_4$ ,  $\eta_1$  by  $\eta_3$ , and  $\eta_2$  by  $\eta_4$ .

#### Test 6

This test is conducted only if  $i = 1$ ; i. e., side four is on the periphery. The test is identical to Test 3 except that  $\xi_1$  is replaced by  $\xi_4$ ,  $\xi_2$  by  $\xi_1$ ,  $\eta_1$  by  $\eta_4$ , and  $\eta_2$  by  $\eta_1$ .

#### Test 7

The final test is to ensure that the peripheral panel is not the previous panel; i. e.,

$$i \neq i_p$$

$$j \neq j_p$$

$$k \neq k_p$$

(175)

where p denotes the previous panel.

If the peripheral panel successfully passes all the required tests, it is accepted as the entrance panel for the streamline.

USAGE:

COMMON (See program MAIN for blank COMMON description.)

COMMON /SL1/ (See subroutine PANEL.)

COMMON /SL2/ (See subroutine PANEL.)

INTEGER EE, EX

CALL SEARCH (I, J, K, L, EX, GX, HX, II, JJ, KK, EE,  
DDE, ITERM)

Input: NETSP = number of source-panel networks

MSP } = arrays containing the dimensions of  
NSP } the source-panel networks

A11  
A12  
A13  
A21 }  
A22 } arrays of the direction cosines of  
A23 } the source panels in the element  
ANX } coordinate system  
ANY }  
ANZ }

XBP } arrays of the source-panel boundary  
YBP } = points in the reference coordinate  
ZBP } system

LPERF = array of codes to indicate a panel on  
the periphery of a network  
= 0, not a peripheral panel  
= 1, a peripheral panel

G1 } arrays of  $\xi$  coordinates of the source-  
G2 } = panel corner points in the element  
G3 } coordinate system  
G4 }

H1 } arrays of  $\eta$  coordinates of the source-  
H2 } = panel corner points in the element  
H3 } coordinate system  
H4 }

**E13** = the boundary-point tolerance,  $\epsilon_{13}$   
**I** = position index of the previous source panel  
**J** = column index of the previous source panel  
**K** = network index of the previous source panel  
**L** = source-panel accumulative number of the previous panel  
**EX** = exit side number of the previous panel  
**GX** =  $\xi$  coordinate of the streamline exit point of the previous panel  
**HX** =  $\eta$  coordinate of the streamline exit point of the previous panel  
**Output:** **II** = position index of the entrance panel  
**JJ** = column index of the entrance panel  
**KK** = network index of the entrance panel  
**EE** = entrance side number of the panel  
**DDE** = fractional distance of the streamline starting point along the entrance side of the panel  
**ITERM** = 0, streamline did not terminate; search was successful  
 = 5, streamline terminated; search was unsuccessful

**SUBPROGRAMS**

**CALLED:** None

**ERROR RETURNS:** None

**RESTRICTIONS:** The previous panel and the peripheral panels must be four-sided.

**SUBJECT:****FORTRAN IV Subroutine SFLOW****PURPOSE:**

To calculate the symmetric flow velocity component matrices ( $[VX_{ij}]$ ,  $[VY_{ij}]$ , and  $[VZ_{ij}]$ ), and to store them in data files

**METHOD:**

A potential-flow model in a symmetric flow field is analyzed by specifying only the right half of the model. The specified side is referred to as the basic side, and its image across the x-z plane is referred to as the reflected side. A matrix element is composed of the velocity components at a boundary point on the basic side due to singularities of unit strength of the basic side and of the reflected side. For example, the x component of velocity is given by

$$VX_{ij} = VX_{ij}(\text{basic}) + VX_{ij}(\text{reflected}) \quad (176)$$

An element of the  $[VX_{ij}]$  matrix is the x component of velocity induced at the  $i$ th boundary point due to the  $j$ th unit strength singularity. Correspondingly, elements in the  $[VY_{ij}]$  and  $[VZ_{ij}]$  matrices contain the y and z components of the velocity. Because of computer memory size (core) limitations, the velocity component matrices must be partitioned. The maximum matrix partition size was chosen to allow storage of three partitions (one from each matrix) in the core at the same time. The scheme used to compute the matrices is shown in Figure 73.

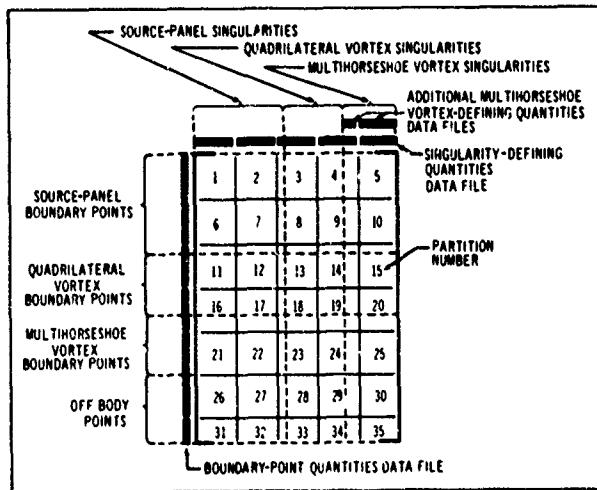


Figure 73. Subroutine SFLOW—Velocity Component Matrix.

Initially, the singularity-defining quantities (DQ) data file, the additional multihorseshoe vortex-defining quantities (MHVDQ) data files, and the boundary-point quantities (BPQ) data file are rewound. Then the BPQ data file is read until the boundary-point quantities for the first row of partitions are in the core, and the DQ data file is read until the singularity-defining quantities for the first column of partitions are also in the core. Now the velocity component elements ( $VX_{ij}$ ,  $VY_{ij}$ ,  $VZ_{ij}$ ) of the first partition (first row, first column) are calculated, stored in the core, and then transferred to data files. Then the defining quantities for the next column of partitions are read, and the elements of the second partition (first row, second column) are calculated. This continues for the remaining column partitions in the first row. When a partition containing multihorseshoe vortex elements is encountered, an MHVDQ data file is read in addition to the DQ data file.

After all the elements of the first row of partitions are stored in the data files, the DQ data file and the MHVDQ data files are rewound. Then the BPQ data file is read until the boundary-point quantities for the second row of partitions are in the core. Now the DQ data file is read just as before, and the elements of the second row of partitions are calculated and stored in data files. This process continues until the elements of the last row of partitions are stored in data files.

Before computing  $VX_{ij}$ ,  $VY_{ij}$ , and  $VZ_{ij}$ , subroutine SFLOW determines the type of singularity. If the singularity is a source panel, SFLOW calls subroutine SOREQ, which calculates the velocity components due to the basic side of the model. The influence of the reflected side is calculated by changing the sign of the y coordinate of the centroid and the signs of the y components of the direction cosines.

$$y_o = -y_o \quad (177)$$

$$a_{12} = -a_{12}$$

$$a_{22} = -a_{22}$$

$$a_{31} = -a_{31}$$

$$a_{33} = -a_{33}$$

(178)

Then SOREQ is called to calculate the velocity components due to the reflected side. The velocity components are

$$\begin{aligned} V_{X_{ij}} &= V_{X_{ij}}(\text{basic}) + V_{X_{ij}}(\text{reflected}) \\ V_{Y_{ij}} &= V_{Y_{ij}}(\text{basic}) + V_{Y_{ij}}(\text{reflected}) \quad (179) \\ V_{Z_{ij}} &= V_{Z_{ij}}(\text{basic}) + V_{Z_{ij}}(\text{reflected}) \end{aligned}$$

If the singularity is a quadrilateral vortex, SFLOW calls subroutine VOREQ, which calculates the velocity components due to the basic and reflected finite line vortices. Subroutine VOREQ is called four times, once for each side of the quadrilateral (see Figure 74).

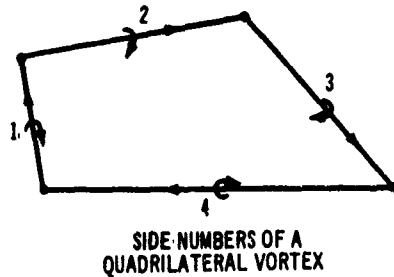


Figure 74. Subroutine SFLOW—Quadrilateral Vortex.

Then SFLOW adds the four parts of each velocity component to obtain the velocity components.

$$\begin{aligned} V_{X_{ij}} &= V_{X_{ij_1}} + V_{X_{ij_2}} + V_{X_{ij_3}} + V_{X_{ij_4}} \\ V_{Y_{ij}} &= V_{Y_{ij_1}} + V_{Y_{ij_2}} + V_{Y_{ij_3}} + V_{Y_{ij_4}} \quad (180) \\ V_{Z_{ij}} &= V_{Z_{ij_1}} + V_{Z_{ij_2}} + V_{Z_{ij_3}} + V_{Z_{ij_4}} \end{aligned}$$

If the singularity is a multihorseshoe vortex, SFLOW calls subroutine VOREQ to calculate the velocity

components due to the basic and reflected line vortices  
(see Figure 75).

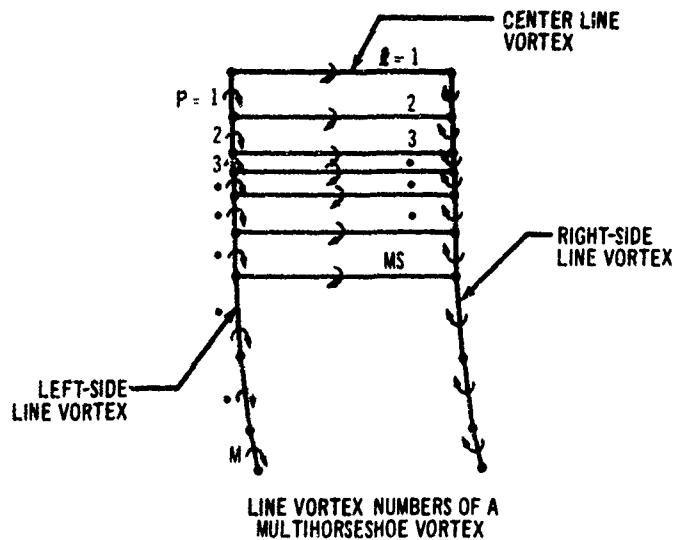


Figure 75. Subroutine SFLOW—Multihorseshoe Vortex.

The velocity components due to the left-side line vortices are given by

$$\begin{aligned}
 Vx_{LS} &= - \sum_{p=1}^{MS-1} WA_p V_{xp} - WA_{MS} \sum_{p=MS}^{M-1} V_{xp} \\
 Vy_{LS} &= - \sum_{p=1}^{MS-1} WA_p V_{yp} - WA_{MS} \sum_{p=MS}^{M-1} V_{yp} \\
 Vz_{LS} &= - \sum_{p=1}^{MS-1} WA_p V_{zp} - WA_{MS} \sum_{p=MS}^{M-1} V_{zp}
 \end{aligned} \quad (181)$$

$$\text{where } WA_p = \sum_{l=1}^p w_l$$

$w_l$  = the weighting factor of the  $l^{\text{th}}$  center line vortex

$\left. \begin{matrix} v_{x_p} \\ v_{y_p} \\ v_{z_p} \end{matrix} \right\}$  = the velocity components of the  $p^{\text{th}}$  left-side line vortex

The velocity components due to the center line vortices are given by

$$v_{x_C} = \sum_{l=1}^{MS} w_l v_{x_l}$$

$$v_{y_C} = \sum_{l=1}^{MS} w_l v_{y_l} \quad (182)$$

$$v_{z_C} = \sum_{l=1}^{MS} w_l v_{z_l}$$

where  $\left. \begin{matrix} v_{x_l} \\ v_{y_l} \\ v_{z_l} \end{matrix} \right\}$  = the velocity of the  $l^{\text{th}}$  center line vortex

The velocity components due to the right-side line vortices are given by

$$v_{x_{RS}} = \sum_{p=1}^{MS-1} WA_p v_{x_p} + WA_{MS} \sum_{p=MS}^{M-1} v_{x_p}$$

$$VY_{RS} = \sum_{p=1}^{MS-1} WA_p V_{y_p} + WA_{MS} \sum_{p=MS}^{M-1} V_{y_p} \quad (183)$$

$$VZ_{RS} = \sum_{p=1}^{MS-1} WA_p V_{z_p} + WA_{MS} \sum_{p=MS}^{M-1} V_{z_p}$$

The velocity components due to the multihorseshoe vortex are found by summing the three contributions

$$\begin{aligned} VX_{ij} &= VX_{LS} + VX_C + VX_{RS} \\ VY_{ij} &= VY_{LS} + VY_C + VY_{RS} \\ VZ_{ij} &= VZ_{LS} + VZ_C + VZ_{RS} \end{aligned} \quad (184)$$

The velocity component matrices are stored in the labeled COMMON block KOM3 until transferred to separate data files by subroutine WRTETP. A description of KOM3 follows.

COMMON /KOM3/ VX(50, 50), VY(50, 50), VZ(50, 50)

where VX = array of one  $\{VX_{ij}\}$  partition

VY = array of one  $\{VY_{ij}\}$  partition

VZ = array of one  $\{VZ_{ij}\}$  partition

When the singularity-defining quantities and boundary-point quantities are read into the core from data files, they are also stored in labeled COMMON blocks. The DQ and BPQ arrays are stored in the COMMON block KOM2 and the MHVDQ arrays are stored in block KOM4. Descriptions of these blocks follow.

COMMON /KOM2/ DQ(40, 50), BP(15, 50)

where DQ = array of singularity-defining quantities for one column of partitions

BP = array of boundary-point quantities for one row of partitions

**COMMON /KOM4/ X(100), Y(100), Z(100), WT(50),  
WA(50), VSX(50), VSY(50), VSZ(50)**

where       $\begin{matrix} X \\ Y \\ Z \end{matrix} \left\{ \right. =$  arrays of corner-point coordinates of  
              one multihorseshoe vortex

WT      = weighting values of the center line  
              vortices of one multihorseshoe vortex

VSX  $\left\{ \begin{matrix} VSY \\ VSZ \end{matrix} \right. =$  arrays used for temporary storage of the  
              velocity components of the right-side,  
              center line, and left-side vortices of one  
              multihorseshoe vortex

**USAGE:**      COMMON (See program MAIN for blank COMMON  
                  description.)

**CALL SFLOW**

Input:      ICOM(2) = maximum matrix partition size

NT8      = data file number of the boundary-  
              point quantities

NT10      = data file number of the singularity-  
              defining quantities

NT11      = data file number of the velocity  
              component matrix  $[VX_{ij}]$

NT12      = data file number of the velocity  
              component matrix  $[VY_{ij}]$

NT13      = data file number of the velocity  
              component matrix  $[VZ_{ij}]$

NT18  $\left\{ \begin{matrix} NT19 \\ NT20 \end{matrix} \right. =$  data file numbers of the additional  
              multihorseshoe vortex-defining  
              quantities

NPR      = number of row partitions in the  
              velocity component matrices

NPC      = number of column partitions in  
              the velocity component matrices

LS      = array containing the number of  
              row (column) elements in each  
              row (column) partition

**SUBPROGRAMS  
CALLED:**

**SOREQ  
VOREQ  
WRTEPP**

**ERROR RETURNS:** None

**RESTRICTIONS:** None

SUBJECT:

FORTRAN IV Subroutine SFLOW1

PURPOSE:

To calculate the symmetric flow velocity components induced at the streamline points by the singularities

METHOD:

The induced velocity components at the  $i^{\text{th}}$  streamline point for the  $k^{\text{th}}$  solution in matrix notation are

$$\begin{aligned} \left[ \begin{matrix} u_{ik} \\ v_{ik} \\ w_{ik} \end{matrix} \right] &= \left[ \begin{matrix} VX_{ij} \\ VY_{ij} \\ VZ_{ij} \end{matrix} \right] \quad \left[ \begin{matrix} \sigma_{jk} \\ \sigma_{jk} \\ \sigma_{jk} \end{matrix} \right] \\ \left( \begin{matrix} VX_{ij} \\ VY_{ij} \\ VZ_{ij} \end{matrix} \right) &= \text{velocity components induced at the } i^{\text{th}} \\ &\quad \text{streamline point due to the } j^{\text{th}} \text{ unit} \\ &\quad \text{strength singularity} \end{aligned} \quad (185)$$

where  $\sigma_{jk}$  = strength of the  $j^{\text{th}}$  singularity  
for the  $k^{\text{th}}$  solution

A potential-flow model in a symmetric flow field is analyzed by specifying only the right half of the model. The specified side is referred to as the basic side, and its image across the  $x$ - $z$  plane is referred to as the reflected side. The velocity components  $VX_{ij}$ ,  $VY_{ij}$ , and  $VZ_{ij}$  are composed of contributions due to singularities of unit strength of the basic side and of the reflected side. The scheme used to compute the elements of the velocity component matrices is shown in Figure 76.

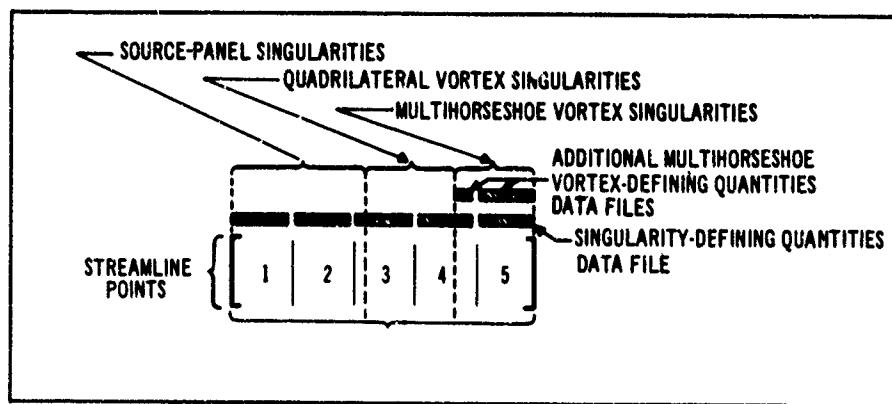


Figure 76. Subroutine SFLOW1—Streamline Velocity Component Matrix.

The streamline points are input through the labeled COMMON block SL3. Initially, the singularity-defining quantities (DQ) data file and the additional multihorseshoe vortex-defining quantities (MHVDQ) data files are rewound.

The DQ data file is read until the singularity-defining quantities for the first partition are in the core. Then the velocity components  $VX_{ij}$ ,  $VY_{ij}$ , and  $VZ_{ij}$  of the first partition are calculated. Then the singularity-defining quantities for the next partition are read in, and the velocity components of the second partition are calculated. This process continues for the remaining partitions. When a partition containing multihorseshoe vortex singularities is encountered, an MHVDQ data file is read in addition to the regular DQ data file.

Before computing  $VX_{ij}$ ,  $VY_{ij}$ , and  $VZ_{ij}$ , subroutine SFLOW1 determines the type of the singularity. If the singularity is a source panel, SFLOW1 calls subroutine SOREQ, which calculates the velocity components due to the basic side of the model. The influence of the reflected side is calculated by changing the sign of the y coordinate of the centroid and the sign of the y components of the direction cosines.

$$y_0 = -y_0 \quad (186)$$

$$a_{12} = -a_{12}$$

$$a_{22} = -a_{22}$$

$$a_{31} = -a_{31}$$

$$a_{33} = -a_{33}$$

(187)

Then SOREQ is called to calculate the velocity components due to the reflected side. The velocity components are

$$\begin{aligned} VX_{ij} &= VX_{ij} (\text{basic}) + VX_{ij} (\text{reflected}) \\ VY_{ij} &= VY_{ij} (\text{basic}) + VY_{ij} (\text{reflected}) \\ VZ_{ij} &= VZ_{ij} (\text{basic}) + VZ_{ij} (\text{reflected}) \end{aligned} \quad (188)$$

If the singularity is a quadrilateral vortex, SFLOW1 calls subroutine VOREQ, which calculates the velocity components due to the basic and reflected finite line vortices. VOREQ is called four times, once for each side of the quadrilateral (see Figure 77).

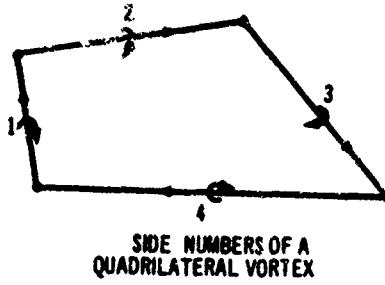


Figure 77. Subroutine SFLOW1—Quadrilateral Vortex.

SFLOW1 adds the four parts of each velocity component to obtain the velocity components.

$$VX_{ij} = VX_{ij_1} + VX_{ij_2} + VX_{ij_3} + VX_{ij_4}$$

$$VY_{ij} = VY_{ij_1} + VY_{ij_2} + VY_{ij_3} + VY_{ij_4} \quad (189)$$

$$VZ_{ij} = VZ_{ij_1} + VZ_{ij_2} + VZ_{ij_3} + VZ_{ij_4}$$

If the singularity is a multihorseshoe vortex, SFLOW1 calls subroutine VOREQ to calculate the velocity components due to the basic and reflected line vortices (see Figure 78).

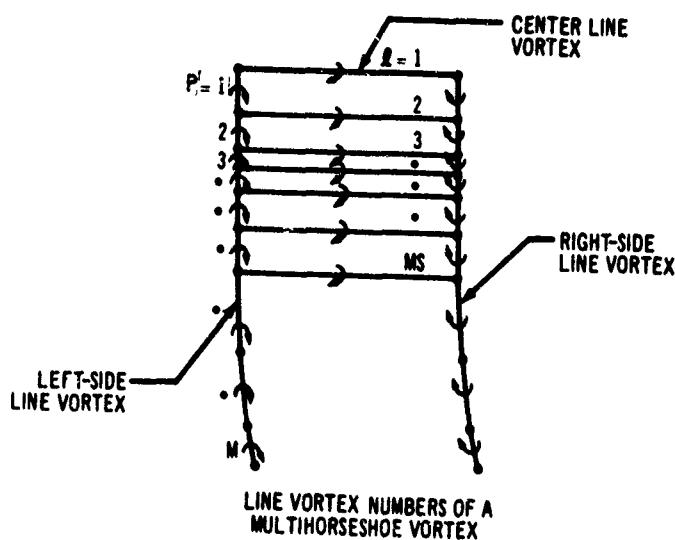


Figure 78. Subroutine SFLOW1—Multihorseshoe Vortex.

The velocity components due to the left-side line vortices are given by

$$V_{x_{LS}} = - \sum_{p=1}^{MS-1} WA_p V_{x_p} - WA_{MS} \sum_{p=MS}^M V_{x_p}$$

$$V_{y_{LS}} = - \sum_{p=1}^{MS-1} WA_p V_{y_p} - WA_{MS} \sum_{p=MS}^M V_{y_p} \quad (190)$$

$$V_{z_{LS}} = - \sum_{p=1}^{MS-1} WA_p V_{z_p} - WA_{MS} \sum_{p=MS}^M V_{z_p}$$

where  $WA_p = \sum_{l=1}^p w_l$

$w_l$  = the weighting value of the  $l^{\text{th}}$  center line vortex

$$\left. \begin{array}{l} V_{x_p} \\ V_{y_p} \\ V_{z_p} \end{array} \right\} = \text{the velocity components of the } p^{\text{th}} \text{ left-side line vortex}$$

The velocity components due to the center line vortices are given by

$$\begin{aligned} V_{x_C} &= \sum_{l=1}^{MS} w_l V_{x_l} \\ V_{y_C} &= \sum_{l=1}^{MS} w_l V_{y_l} \\ V_{z_C} &= \sum_{l=1}^{MS} w_l V_{z_l} \end{aligned} \quad (191)$$

where  $\left. \begin{array}{l} V_{x_l} \\ V_{y_l} \\ V_{z_l} \end{array} \right\} = \text{the velocity components of the } l^{\text{th}} \text{ center line vortex}$

The velocity components due to the right-side line vortices are given by

$$\begin{aligned}
 Vx_{RS} &= \sum_{p=1}^{MS-1} WA_p V_x_p + WA_{MS} \sum_{p=MS}^M V_x_p \\
 Vy_{RS} &= \sum_{p=1}^{MS-1} WA_p V_y_p + WA_{MS} \sum_{p=MS}^M V_y_p \quad (192) \\
 Vz_{RS} &= \sum_{p=1}^{MS-1} WA_p V_z_p + WA_{MS} \sum_{p=MS}^M V_z_p
 \end{aligned}$$

The velocity components due to the multihorseshoe vortex are found by summing the three contributions

$$\begin{aligned}
 Vx_{ij} &= Vx_{LS} + Vx_C + Vx_{RS} \\
 Vy_{ij} &= Vy_{LS} + Vy_C + Vy_{RS} \quad (193) \\
 Vz_{ij} &= Vz_{LS} + Vz_C + Vz_{RS}
 \end{aligned}$$

When the singularity-defining quantities for a partition are read in, they are stored in labeled COMMON blocks. The DQ array is stored in the COMMON block KOM2, and the MHVDQ array is stored in the COMMON block KOM4. Both COMMON blocks are described in subroutine SFLOW. The singularity strength matrix, which is stored one partition at a time in the labeled COMMON block SL5, is read into the core by the matrix routine READTP. The  $\{u_{ik}\}$ ,  $\{v_{ik}\}$ , and  $\{w_{ik}\}$  matrices are also stored in the COMMON block SL5.

**USAGE:**

- COMMON (See program MAIN for blank COMMON description.)
- COMMON /SL3/ (See subroutine AEROSL.)
- COMMON /SL5/ (See subroutine AEROSL.)
- CALL SFLOW1 (KSOL, MSL)

<b>Input:</b>	<b>ICOM(2)</b>	= maximum matrix partition size
	<b>NT1</b>	= data file number of the singularity strength matrix $\{\sigma_{jk}\}$
	<b>NT10</b>	= data file number of the singularity-defining quantities
	<b>NT18 } NT19 } NT20 }</b>	= data file number of the additional multihorseshoe vortex-defining quantities
	<b>NPC</b>	= number of partitions in the $\{\sigma_{jk}\}$ matrix
	<b>LS</b>	= array containing the number of row elements in each partition
	<b>XBO } YBO } ZBO }</b>	= arrays of streamline point coordinates
	<b>KSOL</b>	= solution number (the value of k)
	<b>MSL</b>	= number of streamline points
<b>Output:</b>	<b>VXBO } VYBO } VZBO }</b>	= arrays of the induced velocity components (u, v, w)

**SUBPROGRAMS  
CALLED:**      **SOREQ**  
**VOREQ**  
**READTP**

**ERROR RETURNS:** None

**RESTRICTIONS:** None

SUBJECT:

**FORTRAN IV Subroutine SNF**

PURPOSE:

To store and print the specified normal velocities at a singularity boundary point.

METHOD:

The specified normal velocities at the boundary point ( $U_{s_1}, U_{s_2}, \dots, U_{s_N}$ , where N is the number of simultaneous solutions desired) are read from a data file prepared by subroutine READDP and stored in the boundary-point quantities array. The specified normal velocities override the boundary-point quantities (zero's) previously calculated by subroutine SORDQ or VORDQ. Except where overridden, the boundary-point quantities array remains unchanged. A list of the boundary-point quantities array is given below.

BP(1,JC2)	to		remain unchanged
BP(10,JC2)			
BP(11,JC2)	= $U_{s_1}$	specified normal velocities at the boundary point (maximum of 5)	
BP(12,JC2)	= $U_{s_2}$		
BP(13,JC2)	= $U_{s_3}$		
BP(14,JC2)	= $U_{s_4}$		
BP(15,JC2)	= $U_{s_5}$		

where JC2 denotes the row element number of a partition.

Finally, the specified normal velocities are printed via the output data file. The boundary-point quantities array, BP, is passed to higher-level subroutines through a labeled COMMON statement.

USAGE:

COMMON /COM2/ DQ(40,50), BP(15,50)

CALL SNF (OPTUS, JC2, NT17, NTSOUT)

Input: OPTUS = specified normal velocity option code  
JC2 = row element number of a partition  
NT17 = data file number of the specified normal velocities  
NTSOUT = output data file number

**Output:** BP = boundary-point quantities array (only locations overridden by the specified normal velocities are output)

**SUBPROGRAMS**

**CALLED:** None

**ERROR RETURNS:** None

**RESTRICTIONS:** None

**SUBJECT:**

**FORTRAN IV Subroutine SORDQ**

**PURPOSE:**

To calculate the defining quantities and boundary-point quantities of a source-panel singularity

**METHOD:**

A typical input source-panel is shown in Figure 79.

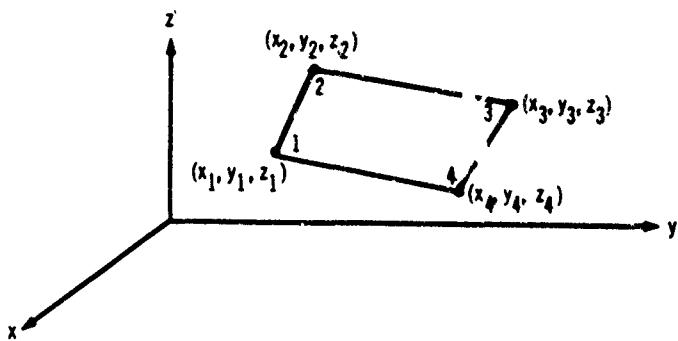


Figure 79. Subroutine SORDQ—Source Panel.

Other allowable panel shapes are shown in Figure 80.

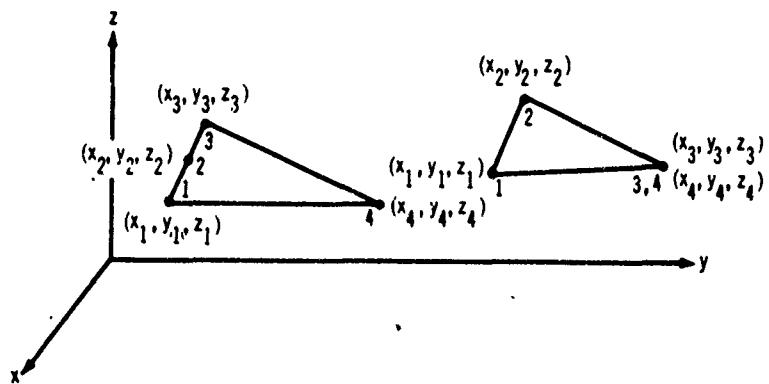


Figure 80. Subroutine SORDQ—Triangular Panels.

All geometry is defined in the reference coordinate system.

The defining quantities are used in the Aerodynamic Section to compute the velocity components induced at some point in space due to the source panel. The velocity components are computed in a coordinate system in the plane of the source panel element and then transformed to the reference coordinate system. SORDQ's first function is to form a plane surface element from the four input corner points. This is accomplished by constructing two diagonal vectors: the vector  $\vec{T}_1$  from point 1 to point 3 and the vector  $\vec{T}_2$  from point 2 to point 4 (see Figure 81).

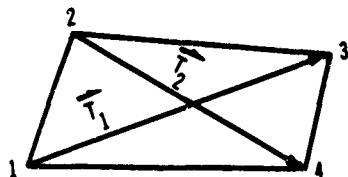


Figure 81. Subroutine SORDQ—Source-Panel Diagonals.

In general, these vectors are not orthogonal. Their components are

$$T_{1x} = x_3 - x_1 \quad T_{1y} = y_3 - y_1 \quad T_{1z} = z_3 - z_1$$

$$T_{2x} = x_4 - x_2 \quad T_{2y} = y_4 - y_2 \quad T_{2z} = z_4 - z_2 \quad (194)$$

The vector  $\vec{N}$  is taken as the cross product of these; i.e.,  $\vec{N} = \vec{T}_2 \times \vec{T}_1$ . Its components are

$$N_x = T_{2y}T_{1z} - T_{2z}T_{1y}$$

$$N_y = T_{2z}T_{1x} - T_{2x}T_{1z} \quad (195)$$

$$N_z = T_{2x}T_{1y} - T_{2y}T_{1x}$$

The components of the unit normal vector,  $\vec{n}$ , are given by

$$n_x = \frac{N_x}{N}$$

$$n_y = \frac{N_y}{N} \quad (196)$$

$$n_z = \frac{N_z}{N}$$

$$\text{where } N = \sqrt{N_x^2 + N_y^2 + N_z^2}$$

The plane of the element is now completely determined if a point in this plane is specified. The average center of the source panel is such a point. Its coordinates are

$$\begin{aligned} x_{ac} &= \frac{1}{4} (x_1 + x_2 + x_3 + x_4) \\ y_{ac} &= \frac{1}{4} (y_1 + y_2 + y_3 + y_4) \\ z_{ac} &= \frac{1}{4} (z_1 + z_2 + z_3 + z_4) \end{aligned} \quad (197)$$

Now the input corner points are projected into the plane of the panel element along the normal vector. The resulting points are the corner points of the planar panel. The signed distance of the  $k^{\text{th}}$  input corner point from the plane is

$$d_k = n_x(x_{ac} - x_k) + n_y(y_{ac} - y_k) + n_z(z_{ac} - z_k), \quad k=1, 2, 3, 4 \quad (198)$$

All the  $d_k$ 's have the same magnitude due to the manner in which the planar panel was generated; those for points 1 and 3 have one sign, and those for points 2 and 4 have the opposite sign. Now the corner-point coordinates of the planar source panel replace the input corner points.

$$\left. \begin{aligned} x_k &= x_k + n_x d_k \\ y_k &= y_k + n_y d_k \\ z_k &= z_k + n_z d_k \end{aligned} \right\}, \quad k = 1, 2, 3, 4 \quad (199)$$

Now the element coordinate system, which requires the components of three mutually perpendicular unit vectors, is constructed. The three unit vectors are: (1) the vector  $\hat{t}_1$ , which is a unit vector parallel to the  $\xi$  axis of the element coordinate system; (2) the unit vector  $\hat{t}_2$ , which is parallel to the  $\eta$  axis; and (3) the unit vector  $\hat{n}$ , which is parallel to the  $\zeta$  axis. The vector  $\hat{t}_1$  is the vector  $T_1$  divided by its length.

$$t_{1x} = \frac{T_{1x}}{T_1}$$

$$t_{1y} = \frac{T_{1y}}{T_1}$$

$$t_{1z} = \frac{T_{1z}}{T_1}$$

(200)

$$\text{where } T_1 = \sqrt{T_{1x}^2 + T_{1y}^2 + T_{1z}^2}$$

The vector  $\hat{t}_2$  is defined to be  $\hat{n} \times \hat{t}_1$ . Its components are

$$t_{2x} = n_y t_{1z} - n_z t_{1y}$$

$$t_{2y} = n_z t_{1x} - n_x t_{1z}$$

$$t_{2z} = n_x t_{1y} - n_y t_{1x}$$

(201)

Direction cosines of the element coordinate system are needed to transform the coordinates of points and the components of vectors from one coordinate system to the other. Symbolically, the direction cosines are given by

$$a_{11} \ a_{12} \ a_{13}$$

$$a_{21} \ a_{22} \ a_{23}$$

$$a_{31} \ a_{32} \ a_{33}$$

$$\text{where } a_{11} = t_{1x} \quad a_{12} = t_{1y} \quad a_{13} = t_{1z}$$

$$a_{21} = t_{2x} \quad a_{22} = t_{2y} \quad a_{23} = t_{2z}$$

$$a_{31} = n_x \quad a_{32} = n_y \quad a_{33} = n_z$$

The direction cosines are used to transform the corner points into the element coordinate system based on the average center point as origin. The coordinates of the corner points are denoted by  $\xi_k^*$ ,  $\eta_k^*$ , and  $\zeta_k^*$  in the element coordinate system, where  $k = 1, 2, 3, 4$ .

Because the corner points lie in the plane of the element,  $\zeta_k^* = 0$ . Also, because of the orientation of the  $t_1$  vector, the coordinate  $\eta_1^*$  and the coordinate  $\eta_3^*$  are equal. The transformation equations are

$$\left. \begin{aligned} \xi_k^* &= a_{11}(x_k - x_{ac}) + a_{12}(y_k - y_{ac}) + a_{13}(z_k - z_{ac}) \\ \eta_k^* &= a_{21}(x_k - x_{ac}) + a_{22}(y_k - y_{ac}) + a_{23}(z_k - z_{ac}) \end{aligned} \right\}, k=1,2,3,4 \quad (202)$$

The origin of the element coordinate system is now transferred to the centroid of the source panel (see Figure 82).

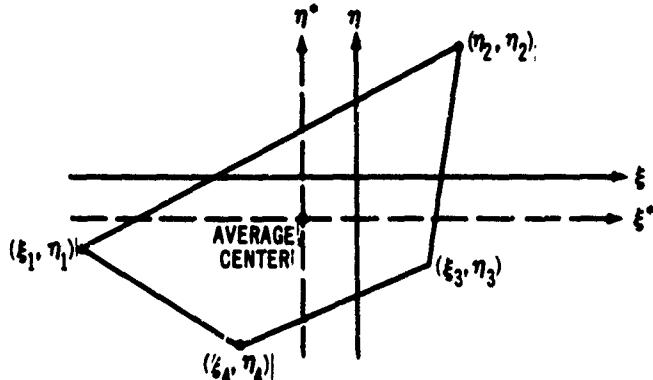


Figure 82. Subroutine SORDQ—Source-Panel Centroid.

The coordinates of the centroid are

$$\begin{aligned} \xi_o &= \frac{1}{3} \frac{1}{\eta_2^* - \eta_4^*} \left[ \xi_4^* (\eta_1^* - \eta_2^*) + \xi_2^* (\eta_4^* - \eta_1^*) \right] \\ \eta_o &= -\frac{1}{3} \eta_1^* \end{aligned} \quad (203)$$

Since the centroid is the origin of the element coordinate system, its coordinates in the reference coordinate system are required in the transformations between coordinate systems. The coordinates are given by

$$x_o = x_{ac} + a_{11}\xi_o + a_{21}\eta_o$$

$$y_0 = y_{ac} + a_{12}\xi_0 + a_{22}\eta_0 \quad (204)$$

$$z_0 = z_{ac} + a_{13}\xi_0 + a_{23}\eta_0$$

By definition, the boundary point of the panel is located at the centroid.

The coordinates of the corner points in the element coordinate system are

$$\left. \begin{aligned} \xi_k &= \xi_k^* - \xi_0 \\ \eta_k &= \eta_k^* - \eta_0 \end{aligned} \right\}, \quad k = 1, 2, 3, 4 \quad (205)$$

The area of the source panel is

$$A = 1/2 (\xi_3 - \xi_1)(\eta_2 - \eta_4) \quad (206)$$

and the second moments are

$$\begin{aligned} I_\xi &= \frac{1}{12} (\xi_3 - \xi_1) \left[ \eta_1 (\xi_4 - \xi_2) (\xi_1 + \xi_3 + \xi_2 + \xi_4) \right. \\ &\quad \left. + (\eta_2 - \eta_4) (\xi_1^2 + \xi_1 \xi_3 + \xi_3^2) \right. \\ &\quad \left. + \xi_2 \eta_2 (\xi_1 + \xi_3 + \xi_2) - \xi_4 \eta_4 (\xi_1 + \xi_3 + \xi_4) \right] \quad (207) \end{aligned}$$

$$\begin{aligned} I_{\xi\eta} &= \frac{1}{24} (\xi_3 - \xi_1) \left[ 2 \xi_4 (\eta_1^2 - \eta_4^2) - 2 \xi_2 (\eta_1^2 - \eta_2^2) \right. \\ &\quad \left. + (\xi_1 + \xi_3) (\eta_2 - \eta_4) (2 \eta_1 + \eta_2 + \eta_4) \right] \quad (208) \end{aligned}$$

$$\begin{aligned} I_\eta &= \frac{1}{12} (\xi_3 - \xi_1) (\eta_2 - \eta_4) \left[ (\eta_1 + \eta_2 + \eta_4)^2 \right. \\ &\quad \left. - \eta_1 (\eta_2 + \eta_4) - \eta_2 \eta_4 \right] \quad (209) \end{aligned}$$

Some additional quantities necessary when the exact formulation of the velocity component is required are the lengths of the panel sides.

$$\begin{aligned} d_1 &= \sqrt{(\xi_2 - \xi_1)^2 + (\eta_2 - \eta_1)^2} \\ d_2 &= \sqrt{(\xi_3 - \xi_2)^2 + (\eta_3 - \eta_2)^2} \\ d_3 &= \sqrt{(\xi_4 - \xi_3)^2 + (\eta_4 - \eta_3)^2} \\ d_4 &= \sqrt{(\xi_1 - \xi_4)^2 + (\eta_1 - \eta_4)^2} \end{aligned} \quad (210)$$

and the slopes of the sides

$$\begin{aligned}m_1 &= \frac{\eta_2 - \eta_1}{\xi_2 - \xi_1} \\m_2 &= \frac{\eta_3 - \eta_2}{\xi_3 - \xi_2} \\m_3 &= \frac{\eta_4 - \eta_3}{\xi_4 - \xi_3} \\m_4 &= \frac{\eta_1 - \eta_4}{\xi_1 - \xi_4}\end{aligned}\tag{211}$$

If an infinite or near-infinite slope is encountered,

$$\Delta \xi_k < 0.000001, \quad k = 1, 2, 3, 4 \tag{212}$$

the slope is not computed and an indicator is turned "on," IS = 1.

The indicator is checked in the evaluation of  $V_\xi$  (see subroutine SOREQ) to exclude the infinite slope side(s) from the computations.

A list of the source-panel-defining quantities array follows:

$$\left. \begin{array}{l} DQ(1,JC2) = a_{11} \\ DQ(2,JC2) = a_{12} \\ DQ(3,JC2) = a_{13} \\ DQ(4,JC2) = a_{21} \\ DQ(5,JC2) = a_{22} \\ DQ(6,JC2) = a_{23} \\ DQ(7,JC2) = a_{31} \\ DQ(8,JC2) = a_{32} \\ DQ(9,JC2) = a_{33} \end{array} \right\} \text{direction cosines of the element coordinate system}$$

$$\left. \begin{array}{l} DQ(10,JC2) = \xi_1 \\ DQ(11,JC2) = \xi_2 \\ DQ(12,JC2) = \xi_3 \\ DQ(13,JC2) = \xi_4 \\ DQ(14,JC2) = \eta_1 \\ DQ(15,JC2) = \eta_2 \\ DQ(16,JC2) = \eta_3 \\ DQ(17,JC2) = \eta_4 \end{array} \right\} \text{coordinates of the source-panel corner points in the element coordinate system}$$

$$\left. \begin{array}{l} DQ(18,JC2) = x_o \\ DQ(19,JC2) = y_o \\ DQ(20,JC2) = z_o \end{array} \right\} \text{coordinates of the source-panel centroid and origin of the element coordinate system in the reference coordinate system}$$

$$DQ(21,JC2) = T \quad \text{longest source panel diagonal, the maximum of } T_1 \text{ and } T_2$$

$$DQ(22,JC2) = A \quad \text{area of source panel}$$

$$\left. \begin{array}{l} DQ(23,JC2) = I_\xi \\ DQ(24,JC2) = I_{\xi\eta} \\ DQ(25,JC2) = I_\eta \end{array} \right\} \text{second moments of source panel}$$

$$\left. \begin{array}{l} DQ(26,JC2) = d_1 \\ DQ(27,JC2) = d_2 \\ DQ(28,JC2) = d_3 \\ DQ(29,JC2) = d_4 \end{array} \right\} \text{lengths of the panel sides}$$

$DQ(30, JC2) = m_1$	slopes of the panel sides
$DQ(31, JC2) = m_2$	
$DQ(32, JC2) = m_3$	
$DQ(33, JC2) = m_4$	
$DQ(34, JC2) = IS$	infinite slope indicator; if 0, no infinite or near-infinite slopes; if 1, one or more infinite or near-infinite slopes encountered
$DQ(35, JC2)$ to $DQ(39, JC2) = 0.0$	not defined
$DQ(40, JC2) = 1.0$	a code identifying this as a source-panel-defining quantities array
where JC2 denotes the <u>column</u> element number of a partition.	
A list of the boundary-point quantities follows:	
$BP(1, JC2) = x_0$	coordinates of the boundary point of the source panel (also the centroid) in the reference coordinate system
$BP(2, JC2) = y_0$	
$BP(3, JC2) = z_0$	
$BP(4, JC2) = n_x$	components of the unit normal vector $\vec{n}$ of the source panel
$BP(5, JC2) = n_y$	
$BP(6, JC2) = n_z$	
$BP(7, JC2) = A$	area of the source panel
$BP(8, JC2) = 0.0$	not defined
$BP(9, JC2) = 0.0$	
$BP(10, JC2) = 1.0$	a code identifying this as a source-panel boundary-point quantities array

$\left. \begin{array}{l} \text{BP}(11, \text{JC2}) \\ \text{to} \\ \text{BP}(15, \text{JC2}) \end{array} \right\} = 0.0$  } normal velocity at the boundary points. These can be overridden by the user of the program (see subroutine SNF).

where JC2 now denotes the row element number of a partition.

The singularity-defining quantities array, DQ, and boundary-point quantities array, BP, are passed to other subroutines through labeled COMMON block COM2.

**USAGE:**

**COMMON/COM2/DQ(40,50), BP(15,50)**

**DIMENSION X(4), Y(4), Z(4)**

**CALL SORDQ (JC2, X, Y, Z, D1)**

**Input:** JC2 = row (column) element number of a partition

X } arrays of the coordinates of the source-  
Y } = panel corner points in the reference  
Z } coordinate system

**Output:** DQ = singularity-defining quantities array

BP = boundary-point quantities array

D1 = the distance,  $d_1$ , between the input corner point 1 of the source panel and its projection into the planar approximation of the panel

**SUBPROGRAMS**

**CALLED:** None

**ERROR RETURNS:** None

**RESTRICTIONS:** The source panel must have a finite area.

**SUBJECT:**

**FORTRAN IV Subroutine SOREQ**

**PURPOSE:**

To calculate the velocity components induced at a point in space by a source-panel singularity of unit strength

**METHOD:**

The velocity components are functions of geometrical quantities and depend on the coordinates of the point in space ( $x, y, z$ ) and on the orientation of the planar source panel. All geometry is defined in the reference coordinate system (see Figure 83).

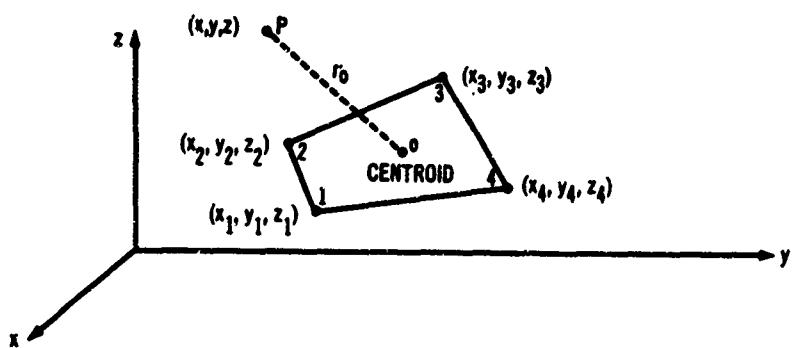


Figure 83. Subroutine SOREQ—Influence of a Source Panel.

The velocity components are calculated in the local coordinate system ( $\xi, \eta, \zeta$ ) of the source panel and then transformed to the reference coordinate system (see Figure 84).  $\xi$  and  $\eta$  are in the plane of the panel and  $\zeta$  is outward from the panel.

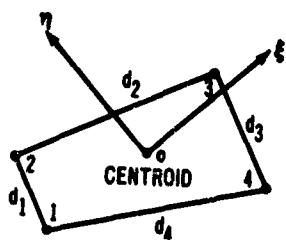


Figure 84. Subroutine SOREQ—Local Coordinate System.

This is called the element coordinate system. The direction cosines of the element coordinate system

$$\begin{matrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{matrix}$$

and the coordinates of the panel centroid ( $x_0, y_0, z_0$ ) in the reference coordinate system are used to transform the coordinates from one system to the other (see subroutine SORDQ for a detailed description). The source-panel geometric quantities necessary to calculate the velocity components are stored in the labeled COMMON singularity-defining quantities array DQ. These quantities are calculated in the Geometric Section of the program by subroutine SORDQ. The coordinates of the point p are given in the reference coordinate system and are transformed to the element coordinate system by

$$\begin{aligned} \xi &= a_{11}(x - x_0) + a_{12}(y - y_0) + a_{13}(z - z_0) \\ \eta &= a_{21}(x - x_0) + a_{22}(y - y_0) + a_{23}(z - z_0) \quad (213) \\ \zeta &= a_{31}(x - x_0) + a_{32}(y - y_0) + a_{33}(z - z_0) \end{aligned}$$

The exact formulas for the velocity components are

$$\begin{aligned} v_\xi &= \frac{\eta_2 - \eta_1}{d_1} \log \left( \frac{r_1 + r_2 - d_1}{r_1 + r_2 + d_1} \right) + \frac{\eta_3 - \eta_2}{d_2} \log \left( \frac{r_2 + r_3 - d_2}{r_2 + r_3 + d_2} \right) + \\ &\quad \frac{\eta_4 - \eta_3}{d_3} \log \left( \frac{r_3 + r_4 - d_3}{r_3 + r_4 + d_3} \right) + \frac{\eta_1 - \eta_4}{d_4} \log \left( \frac{r_4 + r_1 - d_4}{r_4 + r_1 + d_4} \right) \\ v_\eta &= \frac{\xi_1 - \xi_2}{d_1} \log \left( \frac{r_1 + r_2 - d_1}{r_1 + r_2 + d_1} \right) + \frac{\xi_2 - \xi_3}{d_2} \log \left( \frac{r_2 + r_3 - d_2}{r_2 + r_3 + d_2} \right) + \\ &\quad \frac{\xi_3 - \xi_4}{d_3} \log \left( \frac{r_3 + r_4 - d_3}{r_3 + r_4 + d_3} \right) + \frac{\xi_4 - \xi_1}{d_4} \log \left( \frac{r_4 + r_1 - d_4}{r_4 + r_1 + d_4} \right) \quad (214) \end{aligned}$$

$$\begin{aligned}
 v_\xi &= \tan^{-1} \left( \frac{m_1 e_1 - h_1}{\xi r_1} \right) - \tan^{-1} \left( \frac{m_1 e_2 - h_2}{\xi r_2} \right) + \\
 &\quad \tan^{-1} \left( \frac{m_2 e_2 - h_2}{\xi r_2} \right) - \tan^{-1} \left( \frac{m_2 e_3 - h_3}{\xi r_3} \right) + \\
 &\quad \tan^{-1} \left( \frac{m_3 e_3 - h_3}{\xi r_3} \right) - \tan^{-1} \left( \frac{m_3 e_4 - h_4}{\xi r_4} \right) + \\
 &\quad \tan^{-1} \left( \frac{m_4 e_4 - h_4}{\xi r_4} \right) - \tan^{-1} \left( \frac{m_4 e_1 - h_1}{\xi r_1} \right)
 \end{aligned}$$

where  $d_1 = \sqrt{(\xi_2 - \xi_1)^2 + (\eta_2 - \eta_1)^2}$

$$d_2 = \sqrt{(\xi_3 - \xi_2)^2 + (\eta_3 - \eta_2)^2}$$

$$d_3 = \sqrt{(\xi_4 - \xi_3)^2 + (\eta_4 - \eta_3)^2}$$

$$d_4 = \sqrt{(\xi_1 - \xi_4)^2 + (\eta_1 - \eta_4)^2}$$

$$m_1 = \frac{\eta_2 - \eta_1}{\xi_2 - \xi_1} \quad m_2 = \frac{\eta_3 - \eta_2}{\xi_3 - \xi_2}$$

$$m_3 = \frac{\eta_4 - \eta_3}{\xi_4 - \xi_3} \quad m_4 = \frac{\eta_1 - \eta_4}{\xi_1 - \xi_4}$$

and  $r_k = \sqrt{(\xi - \xi_k)^2 + (\eta - \eta_k)^2 + \zeta^2}$ ,  $k = 1, 2, 3, 4$   
 $e_k = \zeta^2 + (\xi - \xi_k)^2$ ,  $k = 1, 2, 3, 4$   
 $h_k = (\eta - \eta_k)(\xi - \xi_k)$ ,  $k = 1, 2, 3, 4$

The defining quantities  $d_1, d_2, d_3, d_4, m_1, m_2, m_3$ , and  $m_4$  are calculated in subroutine SORDQ but are repeated above for clarity.

The velocity component,  $V_\xi$ , depends on the slopes of the panel edges; hence, the slopes must be checked prior to its calculation. This is accomplished by testing a previously defined code, IS. If IS = 0,  $V_\xi$  is calculated as given above. However, IS = 1 indicates that one or more of the slopes are near infinite and that the velocity contribution due to these sides is not computed. Initially,  $V_\xi$  is set equal to zero. If

$$|\xi_2 - \xi_1| < 0.000001 \quad (215)$$

the slope of side 1 is near infinite, and its contribution to the velocity component is not calculated. If on the other hand

$$|\xi_2 - \xi_1| \geq 0.000001 \quad (216)$$

the contribution is computed by

$$V_{\xi 1} = \tan^{-1} \left( \frac{m_1 e_1 - h_1}{t r_1} \right) - \tan^{-1} \left( \frac{m_1 e_2 - h_2}{t r_2} \right) \quad (217)$$

Similarly, the contributions due to sides 2, 3, and 4 are computed. The final velocity component is the sum of the individual contributions

$$V_\xi = V_{\xi 1} + V_{\xi 2} + V_{\xi 3} + V_{\xi 4} \quad (218)$$

A special form of the equation for  $V_\xi$  is required if  $t < 0.000001$ . If the point p is outside the panel,  $V_\xi = 0.0$ .

If the point is inside the panel,  $V_\xi = 2\pi$ .

The exact formulas to represent the velocity due to a source panel are not used when point p is some distance from the singularity. If

$$2.45 \leq \frac{r_o}{t} \leq 4.0 \quad (219)$$

where  $r_o = (x - x_o)^2 + (y - y_o)^2 + (z - z_o)^2$

$t$  = longest source-panel diagonal,

then velocity components are approximated using a multipole expansion about the centroid.

$$\begin{aligned}
 V_\xi &= - \left[ Aw_\xi + \frac{1}{2} I_\xi w_{\xi\xi\xi} + I_\xi \eta w_{\xi\xi\eta} + \frac{1}{2} I_\eta w_{\xi\eta\eta} \right] \\
 V_\eta &= - \left[ Aw_\eta + \frac{1}{2} I_\xi w_{\xi\xi\eta} + I_\xi \eta w_{\xi\eta\eta} + \frac{1}{2} I_\eta w_{\eta\eta\eta} \right] \quad (220) \\
 V_\zeta &= - \left[ Aw_\zeta + \frac{1}{2} I_\xi w_{\xi\xi\zeta} + I_\xi \eta w_{\xi\eta\zeta} + \frac{1}{2} I_\eta w_{\eta\eta\zeta} \right]
 \end{aligned}$$

where

$$\begin{aligned}
 A &= \frac{1}{2} (\xi_3 - \xi_1) (\eta_2 - \eta_4) \\
 w_\xi &= - \xi r_o^{-3} \\
 w_\eta &= - \eta r_o^{-3} \\
 w_\zeta &= - \zeta r_o^{-3} \\
 I_\xi &= \frac{1}{12} (\xi_3 - \xi_1) \left[ \eta_1 (\xi_4 - \xi_2) (\xi_1 + \xi_3 + \xi_2 + \xi_4) + \right. \\
 &\quad (\eta_2 - \eta_4) (\xi_1^2 + \xi_1 \xi_3 + \xi_3^2) + \\
 &\quad \left. \xi_2 \eta_2 (\xi_1 + \xi_3 + \xi_2) - \xi_4 \eta_4 (\xi_1 + \xi_3 + \xi_4) \right] \\
 I_{\xi\eta} &= \frac{1}{24} (\xi_3 - \xi_1) \left[ 2 \xi_4 (\eta_1^2 - \eta_4^2) - 2 \xi_2 (\eta_1^2 - \eta_2^2) + \right. \\
 &\quad \left. (\xi_1 + \xi_3) (\eta_2 - \eta_4) (2 \eta_1 + \eta_2 + \eta_4) \right] \\
 I_\eta &= \frac{1}{12} (\xi_3 - \xi_1) (\eta_2 - \eta_4) \left[ (\eta_1 + \eta_2 + \eta_4)^2 - \right. \\
 &\quad \left. \eta_1 (\eta_2 + \eta_4) - \eta_2 \eta_4 \right] \\
 w_{\xi\xi\xi} &= 3\xi (3p + 10\xi^2) r_o^{-7} \\
 w_{\xi\xi\eta} &= 3\eta p r_o^{-7} \\
 w_{\xi\eta\eta} &= 3\xi q r_o^{-7}
 \end{aligned}$$

$$w_{\eta\eta\eta} = 3\eta(3q + 10\eta^2) r_o^{-7}$$

$$w_{\xi\xi\xi} = 3\xi p r_o^{-7}$$

$$w_{\xi\eta\xi} = -15\xi\eta\xi r_o^{-7}$$

$$w_{\eta\eta\xi} = 3\xi q r_o^{-7}$$

$$p = \eta^2 + \xi^2 - 4\xi^2$$

$$q = \xi^2 + \zeta^2 - 4\eta^2$$

The defining quantities  $A$ ,  $I_\xi$ ,  $I_{\xi\eta}$ , and  $I_\eta$  are calculated in subroutine SORDQ but are repeated above for clarity.

If  $\frac{r_o}{t} > 4.0$ , the velocity components are evaluated directly in the reference coordinate system by the well-known point source formulas.

$$VX = A r_o^{-3} (x - x_o)$$

$$VY = A r_o^{-3} (y - y_o) \quad (221)$$

$$VZ = A r_o^{-3} (z - z_o)$$

If the velocity components were calculated in the element coordinate system, they are transformed to the reference coordinate system using the direction cosines.

$$VX = a_{11} V_\xi + a_{21} V_\eta + a_{31} V_\zeta$$

$$VY = a_{12} V_\xi + a_{22} V_\eta + a_{32} V_\zeta \quad (222)$$

$$VZ = a_{13} V_\xi + a_{23} V_\eta + a_{33} V_\zeta$$

USAGE:

COMMON /KOM2/ DQ(40,50), BP(15,50)

CALL SOREQ (IROW, JCOL, ISYM, X, Y, Z, VX, VY, VZ)

Input: DQ = singularity-defining quantities array  
 IROW } matrix element row and column  
 JCOL } numbers. If IROW = JCOL,  $V_t$  is  
       set equal to  $2\pi$  because the point p  
       is inside the source panel  
 ISYM = a code used to ensure that  $V_t = 2\pi$   
       if IROW = JCOL and the flow is  
       symmetrical  
 $\begin{matrix} X \\ Y \\ Z \end{matrix}$  } reference system coordinates of the  
       point in space p  
 Output: VX } velocity components in the reference  
       VY } coordinate system  
       VZ }

SUBPROGRAMS  
 CALLED: None  
 ERROR RETURNS: None  
 RESTRICTIONS: None

**SUBJECT:** FORTRAN IV Subroutine SPGEO

**PURPOSE:** To calculate the source panel geometric quantities required by the Aerodynamic Section of the program and to store them in partition form in data files

**METHOD:**

The geometric quantities are separated into two arrays: the singularity-defining quantities array and the boundary-point quantities array. Each source panel has a singularity-defining quantities array and a boundary-point quantities array. The two arrays are stored in separate data files for use by the Aerodynamic Section. The singularity-defining quantities data file contains the source-panel-defining quantities and the defining quantities of the other singularities (quadrilateral vortices and multihorseshoe vortices). The source-panel arrays are stored first, followed by the quadrilateral vortex arrays and the multihorseshoe vortex arrays. Each array (the defining quantities for one singularity) is used by the Aerodynamic Section to calculate one column of the aerodynamic velocity components matrices. Because of limited core storage, the quantities are stored as a number of partitioned arrays. This permits the computation of the aerodynamic velocity components as partitions of the entire matrix.

The number of defining quantities arrays in each partition is determined from the COMMON array LS, which contains the number of column elements in each column partition (see subroutine PARTIN). The numbers in the array are used sequentially until the defining quantities of all the singularities are stored in the singularity-defining quantities data file.

The boundary-point quantities data file is formed in the same manner except that the words "row" and "column" are interchanged and the off-body point quantities, if any, are added to the data file immediately following the multihorseshoe vortex boundary-point quantities.

Subroutine SPGEO is concerned only with storing the source-panel geometric quantities in the data files. SPGEO begins by reading all the source-panel corner points of a network from the input data file. Subroutine READBP is called and, if requested, reads in the specified normal velocities of the network panels and stores them in a scratch data file.

Now, SPGEO treats each panel individually. First, the network indices associated with the panel are replaced by partition index numbers (this will help in storing the geometric quantities in partition form in the data files). Then SPGEO calls subroutine SORDQ, which calculates the defining quantities of the panel. If the option to specify the normal

velocity is exercised, a call to subroutine SNF is made, and the specified normal velocities of the panel are placed in the boundary-point quantities array. Finally, selected geometric properties of the panel are output to aid in the detection of the input errors. This procedure is repeated for every source panel of the potential-flow model.

As mentioned previously, the defining quantities and the boundary-point quantities of the source panels are stored in the core until the end of a partition is reached (this condition is determined by the partition index numbers). When this occurs, the quantities are added to the data files. The geometric quantities for the next partition are calculated, stored in the core, and finally added to the data files. If the last source panel does not complete a partition (and it probably will not), the geometric quantities of the partition remain in the core until additional geometric quantities required for the quadrilateral vortices (calculated by subroutine QVGEO) complete the partition. Then the geometric quantities of the two types of singularities are placed in partitioned form on data files uninterrupted by the change in singularities.

The singularity-defining quantities array and boundary-point array are stored in the labeled COMMON block COM2. The corner-point coordinates for the singularities of a network are stored in the labeled COMMON block COM3.

**USAGE:** COMMON (See program MAIN for blank COMMON description.)

COMMON /COM2/ DQ(40, 50), BP(15, 50)

CALL SPGEO (JPC, JC2, JCOL, IRROR)

Input: NTSIN = input data file number  
NTSOUT = output data file number  
NT8 = data file number of the boundary-point quantities  
NT10 = data file number of the singularity-defining quantities  
NT17 = data file number of the specified normal velocities of a network  
NPC = number of row (column)\* partitions in the aerodynamic velocity components matrices

LS	= array containing the number of row (column)* elements in each row (column)* partition
Input/ Output: JPC	= index number of the row (column)* partition
JC2	= index number of the row (column)* element in the row (column)* partition
JCOL	= index number of the row (column)* element in the aerodynamic velocity components matrices
Output: COM(3)	= longest source-panel diagonal
MATSP	= total number of source-panel singularities used in the potential-flow model
MSP } NSP }	= arrays containing the dimensions of the source-panel networks
IRROR	= 0, routine successful = 1, input error in source-panel geometry
DQ	= defining quantities for the last partition containing source panels
BP	= boundary-point quantities for the last partition containing source panels

SUBPROGRAMS

CALLED:

READBP  
SORDQ  
SNF

ERROR RETURNS: If an error occurs, this routine returns to subroutine  
CONTRL where an error message is written and executed  
if passed to the next case.

RESTRICTIONS: None

---

\*"Row" refers to the boundary-point quantities, and "column" refers to the singularity-defining quantities.

**SUBJECT:** FORTRAN IV Subroutine STREAM  
**PURPOSE:** To trace streamlines over the portion of the potential-flow model represented by source panels  
**METHOD:** Before tracing any streamlines, this routine reads the required source-panel-defining quantities from a data file and stores them in the labeled COMMON block SL1. Then every source panel on the periphery of a network is identified by a code that is stored in the LPERF array. For each streamline, additional parameters are read from the input data files. These parameters are: the solution desired, the starting panel indices, the entrance side, the fractional distance along the entrance side of the starting point, the maximum length of the streamline, and the fraction of the longest source-panel diagonal used to form the tolerance  $\epsilon_{10}$ .  
 Subroutine READTP is called to read the singularity-induced velocity components from data files. The resultant velocity components at each source panel are computed by adding the free-stream velocity component to the induced components

$$\begin{aligned}
 V_{x_k} &= u_k + V_{\infty x_k} \\
 V_{y_k} &= v_k + V_{\infty y_k} \\
 V_{z_k} &= w_k + V_{\infty z_k}
 \end{aligned} \tag{223}$$

where  $k$  = the source-panel accumulative number  
 $k$  = the solution number

The resultant velocity components of the source panels are also stored in the labeled COMMON block SL1.

Before a panel is traversed, its streamline characteristics are determined by calling subroutine PANEL. After the characteristics are determined, the entrance point of the panel in the element coordinate system is calculated by

$$\begin{aligned}
 \xi_E &= \frac{d}{d_e} [\xi_{e+1} - \xi_e] + \xi_e \\
 \eta_E &= \frac{d}{d_e} [\eta_{e+1} - \eta_e] + \eta_e
 \end{aligned} \tag{224}$$

where  $\left. \begin{array}{l} \xi_e \\ \eta_e \end{array} \right\}$  = coordinates of the panel corner points  
of the entrance side  
 $e$  = number of the entrance side  
 $d/d_e$  = fractional distance along the entrance  
side of the starting point

The unit normal vector components of the streamline in  
the element coordinate system are given by

$$\begin{aligned} b_x &= \frac{n_y V_z - n_z V_y}{b_m} \\ b_y &= \frac{n_z V_x - n_x V_z}{b_m} \\ b_z &= \frac{n_x V_y - n_y V_x}{b_m} \end{aligned} \quad (225)$$

where

$$b_m = \sqrt{(n_y V_z - n_z V_y)^2 + (n_z V_x - n_x V_z)^2 + (n_x V_y - n_y V_x)^2}$$

$\left. \begin{array}{l} n_x \\ n_y \\ n_z \end{array} \right\}$  = unit normal vector components of the  
panel being traversed

The panel entrance side is examined, and if the streamline cannot enter this side, subroutine FAIL is called to determine the reason. If the streamline can enter this side, the  $d/d_e$  values are computed for each exit side of the panel:

$$d/d_e = \frac{V_\eta (\xi_E - \xi_e) - V_\xi (\eta_E - \eta_e)}{V_\eta (\xi_{e+1} - \xi_e) - V_\xi (\eta_{e+1} - \eta_e)} \quad (226)$$

When a side for which  $0 \leq d/d_e \leq 1$  is found, the values of  $d/d_e$  and  $e$  are used to determine the stream-line exit point; i.e.,

$$\begin{aligned}\xi_X &= \frac{d}{d_e} (\xi_{e+1} - \xi_e) + \xi_e \\ \eta_X &= \frac{d}{d_e} (\eta_{e+1} - \eta_e) + \eta_e\end{aligned}\quad (227)$$

The length of the streamline across the panel is computed as

$$\Delta s = \sqrt{(\xi_X - \xi_E)^2 + (\eta_X - \eta_E)^2} \quad (228)$$

And the accumulative length of the streamline is

$$s = s_p + \Delta s \quad (229)$$

where  $s_p$  is the accumulative length through the previous panels. If the accumulative length exceeds the maximum specified length, the streamline is terminated. The streamline length to the midpoint of the panel is given by

$$s_s = s_p + 0.5 \cdot \Delta s \quad (230)$$

The midpoint coordinates are

$$\begin{aligned}x_s &= x + a_{11} \xi_s + a_{21} \eta_s \\ y_s &= y + a_{12} \xi_s + a_{22} \eta_s \\ z_s &= z + a_{13} \xi_s + a_{23} \eta_s\end{aligned}\quad (231)$$

where  $\begin{matrix} x \\ y \\ z \end{matrix} \} = \text{coordinates of the panel boundary points in the reference coordinate system}$

$\begin{matrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \end{matrix} \} = \text{direction cosines of the point in the element coordinate system}$

$$\xi_s = 0.5 \cdot (\xi_E + \xi_X)$$

$$\eta_s = 0.5 \cdot (\eta_E + \eta_X)$$

The coordinates of the streamline points used to determine the streamline flow properties (see subroutine AEROSL) are given by

$$\begin{aligned}x_{bo} &= x + \epsilon_{10} b_x \\y_{bo} &= y + \epsilon_{10} b_y \\z_{bo} &= z + \epsilon_{10} b_z\end{aligned}\quad (232)$$

where  $\epsilon_{10} = \epsilon \cdot t$

and  $\epsilon$  = a function specified by the user  
 $t$  = longest source-panel diagonal

After the panel has been traversed, the entrance side of the next panel is identified. If the streamline has reached the periphery of a network, subroutine SEARCH is called to determine the entrance side of the next panel.

When a streamline is terminated, subroutine ENDSL is called to print out the cause of termination. Then subroutine AEROSL is called to calculate the flow properties along the streamline.

The streamline parameters of the next streamline are read from the input data file and the process is repeated.

USAGE:

COMMON (See program MAIN for blank COMMON description.)

CALL STREAM

Input: ICOM(2) = maximum matrix partition size  
ICOM(8) = number of streamlines  
COM(3) = longest source-panel diagonal  
NT2 = data file number of the x component of the induced velocity, u  
NT3 = data file number of the y component of the induced velocity, v  
NT4 = data file number of the z component of the induced velocity, w  
NTSIN = input data file number  
NTSOUT = output data file number

NT10 = data file number of the singularity-defining quantities  
NPC = number of column partitions in the velocity component matrices  
LS = array containing number of velocity components in each column partition  
NETSP = number of source-panel networks  
MSP } = arrays containing the dimensions of the source-panel networks  
NSP }  
NRSH = number of simultaneous solutions  
FSVX } = arrays of the free-stream velocity components,  $V_{\infty x}$ ,  $V_{\infty y}$ , and  $V_{\infty z}$   
FSVY }  
FSVZ }

SUBPROGRAMS

CALLED:

READTP  
PANEL  
FAIL  
ENDSL  
SEARCH  
AEROSL

ERROR RETURNS: Tracing of the streamlines is terminated if the subroutine is unable to read the defining quantities data file.

RESTRICTIONS: All source panels traversed and panels on the network periphery must be four-sided.

**SUBJECT:****FORTRAN IV Subroutine UFLOW****PURPOSE:**

To calculate the unsymmetric flow velocity component matrices  $[VX_{ij}]$ ,  $[VY_{ij}]$ , and  $[VZ_{ij}]$ , and to store them in data files

**METHOD:**

A potential-flow model in an unsymmetric flow field is analyzed by specifying the entire model.

An element of the  $[VX_{ij}]$  matrix is the x component of velocity induced at the  $i^{\text{th}}$  boundary point due to the  $j^{\text{th}}$  unit strength singularity. Corresponding elements in the  $[VY_{ij}]$  and  $[VZ_{ij}]$  matrices contain the y and z components of the velocity. Because of computer memory size (core) limitations, the velocity component matrices must be partitioned. The maximum matrix partition size was chosen to allow storage of three partitions (one from each matrix) in the core at the same time. The scheme used to compute the matrices is shown in Figure 85.

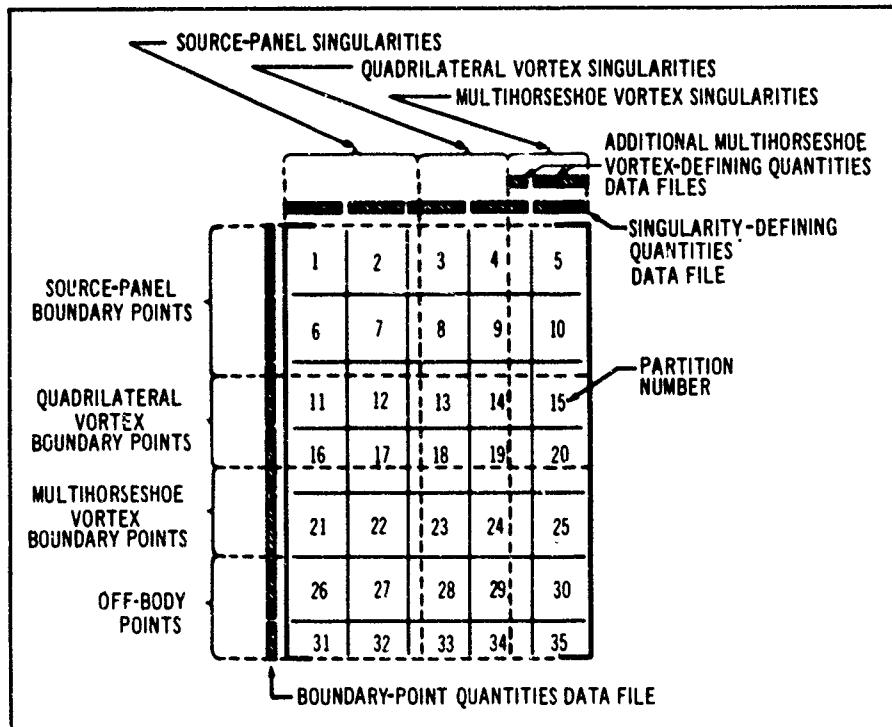


Figure 85. Subroutine UFLOW—Velocity Component Matrix.

Initially, the singularity-defining quantities (DQ) data file, the additional multihorseshoe vortex-defining quantities (MHVDQ) data files, and the boundary-point quantities (BPQ) data file are rewound. Then the BPQ data file is read until the boundary-point quantities for the first row of partitions are in the core, and the DQ data file is read until the singularity-defining quantities for the first column of partitions are also in the core. Now the velocity component elements ( $VX_{ij}$ ,  $VY_{ij}$ ,  $VZ_{ij}$ ) of the first partition (first row, first column) are calculated, stored in the core, and then transferred to data files. The defining quantities for the next column of partitions are read, and the elements of the second partition (first row, second column) are calculated. This continues for the remaining column partitions in the first row. When a partition containing multihorseshoe vortex elements is encountered, an MHVDQ data file is read in addition to the DQ data file.

After all the elements of the first row of partitions are stored in the data files, the DQ data file and the MHVDQ data files are rewound. Then the BPQ data file is read until the boundary-point quantities for the second row of partitions are in the core. Now the DQ data file is read just as before, and the elements of the second row of partitions are calculated and stored in data files. This continues until the elements of the last row of partitions are stored in data files.

Before computing  $VX_{ij}$ ,  $VY_{ij}$ , and  $VZ_{ij}$ , subroutine UFLOW determines the type of singularity. If the singularity is a source panel, UFLOW calls subroutine SOREQ, which calculates the velocity components.

If the singularity is a quadrilateral vortex, UFLOW calls subroutine VOREQ, which calculates the velocity components of a finite line vortex. VOREQ is called four times, once for each side of the quadrilateral (see Figure 86).

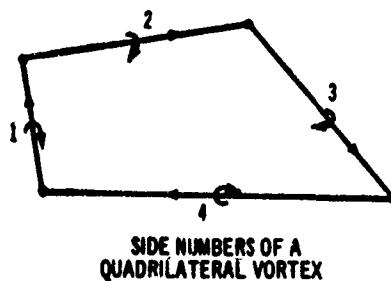


Figure 86. Subroutine UFLOW—Quadrilateral Vortex.

Then UFLOW adds the four parts of each velocity component to obtain the velocity components.

$$VX_{ij} = VX_{ij_1} + VX_{ij_2} + VX_{ij_3} + VX_{ij_4}$$

$$VY_{ij} = VY_{ij_1} + VY_{ij_2} + VY_{ij_3} + VY_{ij_4} \quad (233)$$

$$VZ_{ij} = VZ_{ij_1} + VZ_{ij_2} + VZ_{ij_3} + VZ_{ij_4}$$

If the singularity is a multihorseshoe vortex, UFLOW calls subroutine VOREQ to calculate the velocity components of the line vortices (see Figure 87).

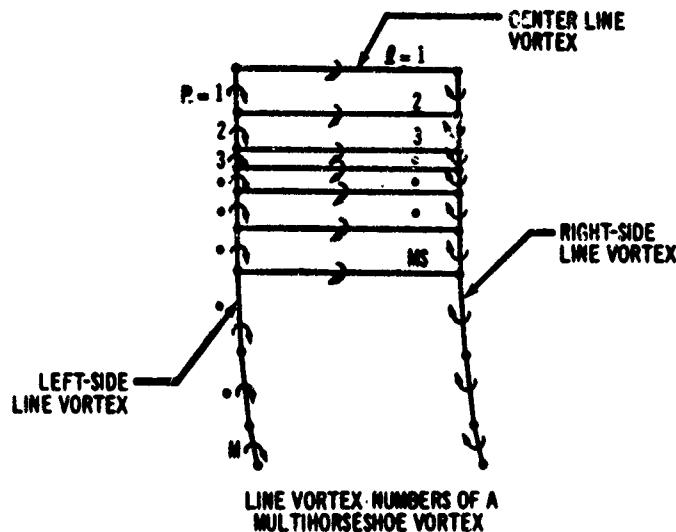


Figure 87. Subroutine UFLOW— Multihorseshoe Vortex.

The velocity components due to the left-side line vortices are given by

$$\begin{aligned} Vx_{LS} &= - \sum_{p=1}^{MS-1} WA_p v_{xp} - WA_{MS} \sum_{p=MS}^M v_{xp} \\ Vy_{LS} &= - \sum_{p=1}^{MS-1} WA_p v_{yp} - WA_{MS} \sum_{p=MS}^M v_{yp} \\ Vz_{LS} &= - \sum_{p=1}^{MS-1} WA_p v_{zp} - WA_{MS} \sum_{p=MS}^M v_{zp} \end{aligned} \quad (234)$$

where  $WA_p = \sum_{\ell=1}^p w_\ell$

$w_\ell$  = the weighting value of the  $\ell^{\text{th}}$  center line vortex

$\left. \begin{matrix} v_{xp} \\ v_{yp} \\ v_{zp} \end{matrix} \right\}$  = the velocity components of the  $p^{\text{th}}$  left-side line vortex

The velocity components due to the center line vortices are given by

$$\begin{aligned} Vx_C &= \sum_{\ell=1}^{MS} w_\ell \cdot v_{x_\ell} \\ Vy_C &= \sum_{\ell=1}^{MS} w_\ell \cdot v_{y_\ell} \\ Vz_C &= \sum_{\ell=1}^{MS} w_\ell \cdot v_{z_\ell} \end{aligned} \quad (235)$$

where  $\left. \begin{matrix} v_{x_\ell} \\ v_{y_\ell} \\ v_{z_\ell} \end{matrix} \right\}$  = the velocity components of the  $\ell^{\text{th}}$  center line vortex

The velocity components due to the right-side line vortices are given by

$$\begin{aligned} Vx_{RS} &= \sum_{p=1}^{MS-1} WA_p V_{xp} + WA_{MS} \sum_{p=MS}^M V_{xp} \\ Vy_{RS} &= \sum_{p=1}^{MS-1} WA_p V_{yp} + WA_{MS} \sum_{p=MS}^M V_{yp} \quad (236) \\ Vz_{RS} &= \sum_{p=1}^{MS-1} WA_p V_{zp} + WA_{MS} \sum_{p=MS}^M V_{zp} \end{aligned}$$

The velocity components due to the multihorseshoe vortex are found by summing the three contributions

$$\begin{aligned} Vx_{ij} &= Vx_{LS} + Vx_C + Vx_{RS} \\ Vy_{ij} &= Vy_{LS} + Vy_C + Vy_{RS} \quad (237) \\ Vz_{ij} &= Vz_{LS} + Vz_C + Vz_{RS} \end{aligned}$$

UFLOW uses the labeled COMMON block KOM3 to store the matrix partitions. The singularity geometric quantities are stored in the labeled COMMON block KOM2 and the additional multihorseshoe geometric quantities are stored in the labeled COMMON block KOM4. The three COMMON blocks are described in subroutine SFLOW.

**USAGE:** COMMON (See program MAIN for blank COMMON description.)

#### CALL UFLOW

Input:	ICOM(2)	= maximum matrix partition size
	NT8	= data file number of the boundary-point quantities
	NT10	= data file number of the singularity-defining quantities
	NT11	= data file number of the velocity component matrix [Vx <sub>ij</sub> ]
	NT12	= data file number of the velocity component matrix [Vy <sub>ij</sub> ]

NT13	= data file number of the velocity component matrix $\{VZ_{ij}\}$
NT18 NT19 NT20	} = data file numbers of the additional multihorseshoe vortex-defining quantities
NPR	= number of row partitions in the velocity component matrices
NPC	= number of column partitions in the velocity component matrices
LS	= array containing the number of row (column) elements in each row (column) partition

**SUBPROGRAMS**

**CALLED:**

SOREQ  
VOREQ  
WRTETP

**ERROR RETURNS:** None

**RESTRICTIONS:** None

SUBJECT: FORTRAN IV Subroutine UFLOW1

PURPOSE: To calculate the unsymmetric flow velocity components induced at the streamline points by the singularities

METHOD: The induced velocity components at the  $i^{\text{th}}$  streamline point for the  $k^{\text{th}}$  solution in matrix notation are

$$\begin{aligned} \left[ u_{ik} \right] &= \left[ Vx_{ij} \right] \left[ \sigma_{jk} \right] \\ \left[ v_{ik} \right] &= \left[ Vy_{ij} \right] \left[ \sigma_{jk} \right] \\ \left[ w_{ik} \right] &= \left[ Vz_{ij} \right] \left[ \sigma_{jk} \right] \end{aligned} \quad (238)$$

where  $\left. \begin{matrix} Vx_{ij} \\ Vy_{ij} \\ Vz_{ij} \end{matrix} \right\}$  = velocity components induced at the  $i^{\text{th}}$  streamline point due to the  $j^{\text{th}}$  unit strength singularity  
 $\sigma_{jk}$  = strength of the  $j^{\text{th}}$  singularity for the  $k^{\text{th}}$  solution

A potential-flow model in an unsymmetric flow field is analyzed by specifying the entire model. The scheme used to compute the elements of the velocity component matrices is shown in Figure 88.

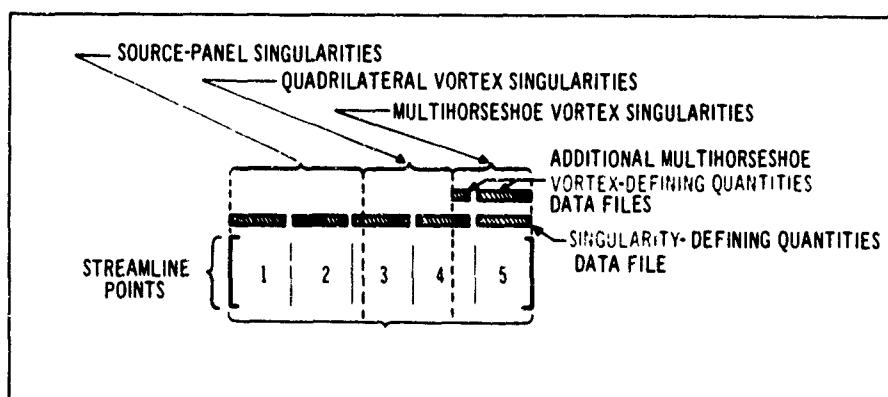


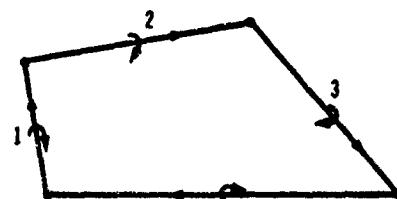
Figure 88. Subroutine UFLOW1—Streamline Velocity Component Matrix.

The streamline points are input through the labeled COMMON block SL3. Initially, the singularity-defining quantities (DQ) data file and the additional multihorseshoe vortex-defining quantities (MHVDQ) data files are rewound.

The DQ data file is read until the singularity-defining quantities for the first partition are in core. Then the velocity components ( $VX_{ij}$ ,  $VY_{ij}$ ,  $VZ_{ij}$ ) of the first partition are calculated. Then the singularity-defining quantities for the next partition are read in, and the velocity components of the second partition are calculated. This process continues for the remaining partitions. When a partition containing multihorseshoe vortex singularities is encountered, an MHVDQ data file is read in addition to the regular DQ data file.

Before computing  $VX_{ij}$ ,  $VY_{ij}$ , and  $VZ_{ij}$ , subroutine UFLOW1 determines the type of the singularity. If the singularity is a source panel, UFLOW1 calls subroutine SOREQ, which calculates the velocity components.

If the singularity is a quadrilateral vortex, UFLOW1 calls subroutine VOREQ, which calculates the velocity components of a finite line vortex. VOREQ is called four times, once for each side of the quadrilateral (see Figure 89).



SIDE NUMBERS OF A  
QUADRILATERAL VORTEX

Figure 89. Subroutine UFLOW1—Quadrilateral Vortex.

UFLOW1 adds the four parts of each velocity component to obtain the velocity components.

$$\begin{aligned} VX_{ij} &= VX_{ij_1} + VX_{ij_2} + VX_{ij_3} + VX_{ij_4} \\ VY_{ij} &= VY_{ij_1} + VY_{ij_2} + VY_{ij_3} + VY_{ij_4} \quad (239) \\ VZ_{ij} &= VZ_{ij_1} + VZ_{ij_2} + VZ_{ij_3} + VZ_{ij_4} \end{aligned}$$

If the singularity is a multihorseshoe vortex, UFLOW1 calls subroutine VOREQ to calculate the velocity components of the line vortices (see Figure 90).

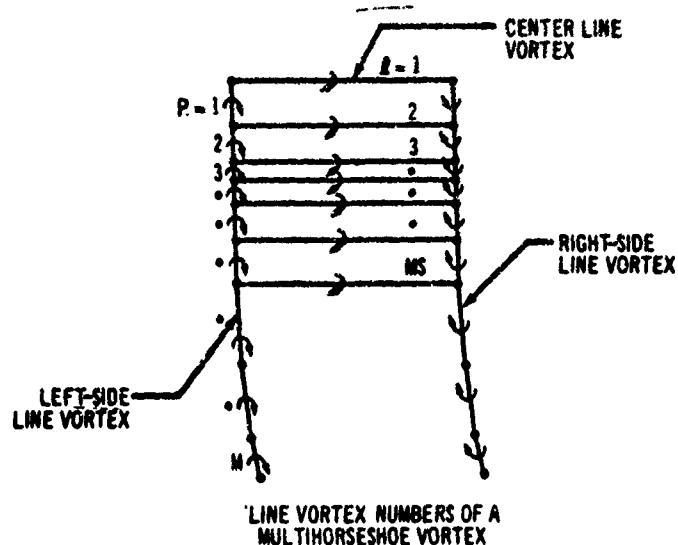


Figure 90. Subroutine UFLOW1— Multihorseshoe Vortex.

The velocity components due to the left-side line vortices are given by

$$\begin{aligned} V_{X_{LS}} &= - \sum_{p=1}^{MS-1} W_{Ap} V_{x_p} - W_{AM} \sum_{p=MS}^M V_{x_p} \\ V_{Y_{LS}} &= - \sum_{p=1}^{MS-1} W_{Ap} V_{y_p} - W_{AM} \sum_{p=MS}^M V_{y_p} \quad (240) \end{aligned}$$

$$V_{Z_{LS}} = - \sum_{p=1}^{MS-1} W_{Ap} V_{z_p} - W_{AM} \sum_{p=MS}^M V_{z_p}$$

where  $W_{Ap} = \sum_{l=1}^p w_l$

$w_l$  = the weighting value of the  $l^{\text{th}}$  center line vortex

$$\left. \begin{array}{l} v_{x_p} \\ v_{y_p} \\ v_{z_p} \end{array} \right\} = \text{the velocity components of the } p^{\text{th}} \text{ left-side line vortex}$$

The velocity components due to the center line vortices are given by

$$\begin{aligned} v_{x_c} &= \sum_{\ell=1}^{MS} w_{\ell} v_{x_{\ell}} \\ v_{y_c} &= \sum_{\ell=1}^{MS} w_{\ell} v_{y_{\ell}} \\ v_{z_c} &= \sum_{\ell=1}^{MS} w_{\ell} v_{z_{\ell}} \end{aligned} \quad (241)$$

$$\text{where } \left. \begin{array}{l} v_{x_{\ell}} \\ v_{y_{\ell}} \\ v_{z_{\ell}} \end{array} \right\} = \text{the velocity components of the } \ell^{\text{th}} \text{ center line vortex}$$

The velocity components due to the right-side line vortices are given by

$$\begin{aligned} v_{x_{RS}} &= \sum_{p=1}^{MS-1} WA_p v_{x_p} + WA_{MS} \sum_{p=MS}^M v_{x_p} \\ v_{y_{RS}} &= \sum_{p=1}^{MS-1} WA_p v_{y_p} + WA_{MS} \sum_{p=MS}^M v_{y_p} \\ v_{z_{RS}} &= \sum_{p=1}^{MS-1} WA_p v_{z_p} + WA_{MS} \sum_{p=MS}^M v_{z_p} \end{aligned} \quad (242)$$

The velocity components due to the multihorseshoe vortex are found by summing the three contributions

$$\begin{aligned}VX_{ij} &= VX_{LS} + VX_C + VX_{RS} \\VY_{ij} &= VY_{LS} + VY_C + VY_{RS} \\VZ_{ij} &= VZ_{LS} + VZ_C + VZ_{RS}\end{aligned}\quad (243)$$

When the singularity-defining quantities for a partition are read in, they are stored in labeled COMMON blocks. The DQ array is stored in the COMMON block KOM2, and the MHVDQ array is stored in the COMMON block KOM4. Both COMMON blocks are described in subroutine SFLOW. The singularity strength matrix, which is stored one partition at a time in the labeled COMMON block SL5, is read into the core by the matrix routine READTP. The  $\{u_{jk}\}$ ,  $\{v_{jk}\}$ , and  $\{w_{jk}\}$  matrices are also stored in the COMMON block SL5.

**USAGE:**

COMMON (See program MAIN for blank COMMON description.)

COMMON /SL3/ (See subroutine AEROSL.)

COMMON /SL5/ (See subroutine AEROSL.)

CALL UFLOW1 (KSOL, MSL)

Input:    ICOM(2)    = maximum matrix partition size  
            NT1       = data file number of the singularity strength matrix  $\{\sigma_{jk}\}$   
            NT10      = data file number of the singularity-defining quantities  
            NT18      { = data file number of the additional  
            NT19      } = multihorseshoe vortex-defining quantities  
            NT20      }  
            NPC       = number of partitions in the  $\{\sigma_{jk}\}$  matrix  
            LS        = array containing the number of row elements in each partition

XBO	{	= arrays of streamline point
YBO		coordinates
ZBO		
KSOL	= solution number (the value of k)	
MSL	= number of streamline points	
Output:	{	= arrays of the induced velocity
VXBO		components (u, v, w)
VYBO		
VZBO		

SUBPROGRAMS  
CALLED:

SOREQ  
VOREQ  
READTP

ERROR RETURNS: None

RESTRICTIONS: None

**SUBJECT:**

**FORTRAN IV Subroutine VORDQ**

**PURPOSE:**

To calculate the defining quantities and boundary-point quantities of a quadrilateral vortex singularity

**METHOD:**

A typical input quadrilateral vortex is shown in Figure 91.

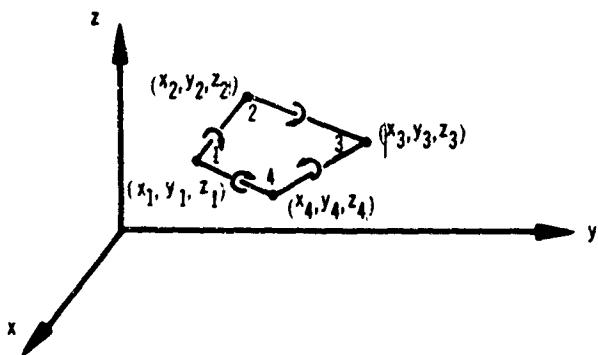


Figure 91. Subroutine VORDQ—Quadrilateral Vortex.

All input geometry is defined in the reference coordinate system. Other allowable quadrilateral shapes are shown in Figure 92.

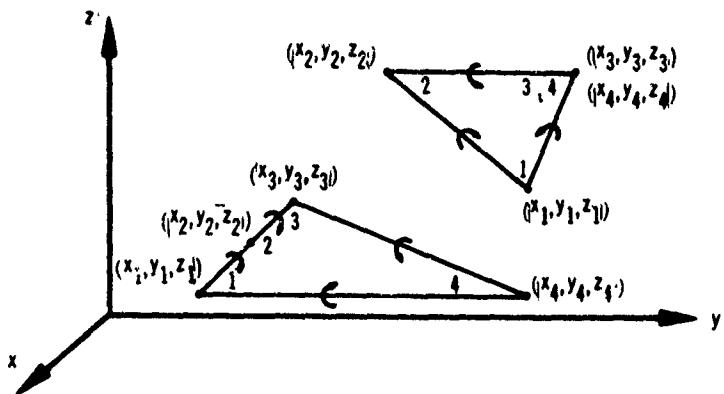


Figure 92. Subroutine VORDQ—Triangular Vortices.

The defining quantities of a quadrilateral vortex require no computations. A list of the singularity-defining quantities array follows:

DQ(1,JC2) = not defined by this subroutine

DQ(2,JC2) =  $x_1$   
 DQ(3,JC2) =  $y_1$   
 DQ(4,JC2) =  $z_1$

coordinates of corner point 1

DQ(5,JC2) =  $x_2$   
 DQ(6,JC2) =  $y_2$   
 DQ(7,JC2) =  $z_2$

coordinates of corner point 2

DQ(8,JC2) =  $x_3$   
 DQ(9,JC2) =  $y_3$   
 DQ(10,JC2) =  $z_3$

coordinates of corner point 3

DQ(11,JC2) =  $x_4$   
 DQ(12,JC2) =  $y_4$   
 DQ(13,JC2) =  $z_4$

coordinates of corner point 4

DQ(14,JC2)  
 to       = 0.0  
 DQ(39,JC2)

not defined

DQ(40,JC2) = 2.0 a code identifying this as a quadrilateral vortex-defining quantities array

where JC2 denotes the column element number of a partition.

The average center of the quadrilateral is used as the location of the boundary point.

$$x = 1/4 (x_1 + x_2 + x_3 + x_4)$$

$$y = 1/4 (y_1 + y_2 + y_3 + y_4)$$

$$z = 1/4 (z_1 + z_2 + z_3 + z_4)$$

(244)

To obtain the unit normal vector, two diagonal vectors are needed: the vector  $\vec{T}_1$  from point 1 to point 3 and the vector  $\vec{T}_2$  from point 2 to point 4. In general, these vectors are not orthogonal. Their components are given by

$$\begin{aligned} T_{1x} &= x_3 - x_1 & T_{1y} &= y_3 - y_1 & T_{1z} &= z_3 - z_1 \\ T_{2x} &= x_4 - x_2 & T_{2y} &= y_4 - y_2 & T_{2z} &= z_4 - z_2 \end{aligned} \quad (245)$$

The vector  $\vec{N}$  is taken as the cross product of these; i.e.,  $\vec{N} = \vec{T}_2 \times \vec{T}_1$ . Its components are

$$\begin{aligned} N_x &= T_{2y} T_{1z} - T_{2z} T_{1y} \\ N_y &= T_{2z} T_{1x} - T_{2x} T_{1z} \\ N_z &= T_{2x} T_{1y} - T_{2y} T_{1x} \end{aligned} \quad (246)$$

The unit normal vector components are computed as

$$\begin{aligned} n_x &= \frac{N_x}{N} \\ n_y &= \frac{N_y}{N} \\ n_z &= \frac{N_z}{N} \end{aligned} \quad (247)$$

$$\text{where } N = \sqrt{N_x^2 + N_y^2 + N_z^2}$$

The area of the quadrilateral is

$$A = \frac{N}{2} \quad (248)$$

A list of the boundary-point quantities array follows:

$$\left. \begin{array}{l} \text{BP}(1,\text{JC2}) = x \\ \text{BP}(2,\text{JC2}) = y \\ \text{BP}(3,\text{JC2}) = z \end{array} \right\} \text{coordinates of the boundary point}$$

$$\left. \begin{array}{l} \text{BP}(4,\text{JC2}) = n_x \\ \text{BP}(5,\text{JC2}) = n_y \\ \text{BP}(6,\text{JC2}) = n_z \end{array} \right\} \begin{array}{l} \text{unit normal vector components of} \\ \text{the quadrilateral} \end{array}$$

$$\text{BP}(7,\text{JC2}) = A \quad \text{area of the quadrilateral}$$

$\begin{array}{l} \text{BP}(8, \text{JC2}) = 0.0 \\ \text{BP}(9, \text{JC2}) = 0.0 \end{array} \left. \begin{array}{l} \text{not defined} \end{array} \right\}$   
 $\text{BP}(10, \text{JC2}) = 2.0$  a code identifying this as a quadrilateral vortex boundary point quantities array  
 $\begin{array}{ll} \text{BP}(11, \text{JC2}) & \text{specified normal velocities at} \\ \text{to} & \text{the boundary point. (These can} \\ \text{BP}(15, \text{JC2}) & \text{be overridden by the user of the} \\ & \text{program; see subroutine SNF.)} \end{array} \left. \begin{array}{l} \text{to} \\ = 0.0 \end{array} \right\}$

where JC2 now denotes the row element number of a partition.

The singularity-defining quantities array DQ and boundary-point quantities array BP are passed to other subroutines through a labeled COMMON statement.

**USAGE:**      COMMON /COM2/ DQ(40, 50), BP(15, 50)

                  DIMENSION X(4), Y(4), Z(4)

                  CALL VORDQ (JC2, X, Y, Z)

Input:    JC2 = row (column) element number of a partition

$\begin{array}{l} \text{X} \\ \text{Y} \\ \text{Z} \end{array} \left. \begin{array}{l} \text{coordinates of the quadrilateral vortex} \\ \text{corner points} \end{array} \right\}$

Output:    DQ = singularity-defining quantities array  
                BP = boundary-point quantities array

**SUBPROGRAMS**

**CALLED:**      None

**ERROR RETURNS:**      None

**RESTRICTIONS:**      The quadrilateral vortex must have area.

**SUBJECT:****FORTRAN IV Subroutine VOREQ****PURPOSE:**

To calculate the velocity components induced at a point in space by a finite line vortex. For symmetric flow, the velocity components also include the influence of the vortex image across the x-z plane.

**METHOD:**

The velocity components, which are functions of geometrical quantities, depend on the coordinates of the point in space ( $x, y, z$ ) and on the coordinates of the end points of the line vortex  $(x_1, y_1, z_1)$  and  $(x_2, y_2, z_2)$ . All geometry is defined in the reference coordinate system (see Figure 93).

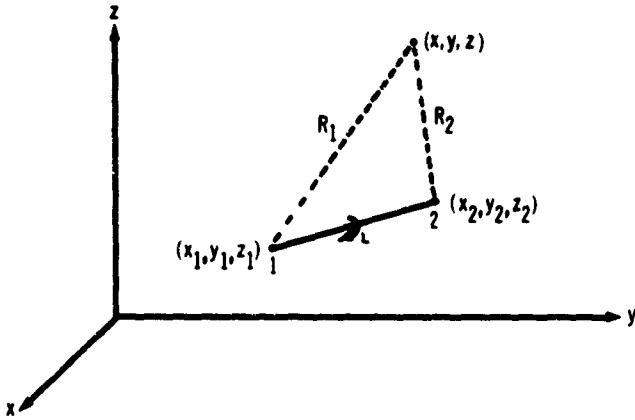


Figure 93. Subroutine VOREQ—Influence of a Line Vortex.

The velocity components induced at a point due to a single line vortex are given by

$$\begin{aligned} V_x &= A\mu \\ V_y &= B\mu \\ V_z &= C\mu \end{aligned} \tag{249}$$

$$\text{where } A = (y_2 - y_1)(z - z_1) - (z_2 - z_1)(y - y_1)$$

$$B = (z_2 - z_1)(x - x_1) - (x_2 - x_1)(z - z_1)$$

$$C = (x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1)$$

$$\mu = \frac{1}{8\pi} \left[ \frac{(2L \cos \theta_1) - (2L \cos \theta_2)}{A^2 + B^2 + C^2} \right]$$

$$2L \cos \theta_1 = \frac{R_1^2 - R_2^2 + L^2}{R_1}$$

$$2L \cos \theta_2 = \frac{R_1^2 - R_2^2 - L^2}{R_2}$$

$$R_1 = \sqrt{(x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2}$$

$$R_2 = \sqrt{(x - x_2)^2 + (y - y_2)^2 + (z - z_2)^2}$$

$$L = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

If

$$A^2 + B^2 + C^2 \leq 1 \times 10^{-20} \quad (250)$$

then

$$\left. \begin{array}{l} v_x \\ v_y \\ v_z \end{array} \right\} = 0.0 \quad (251)$$

For symmetric flow, the velocity components due to the image line vortex of the left half or reflected side of the potential-flow model are computed in addition to the velocity components due to the line vortex of the right half or basic side of the model. This is accomplished by reflecting the point in space across the x-z plane, as shown in Figure 94.

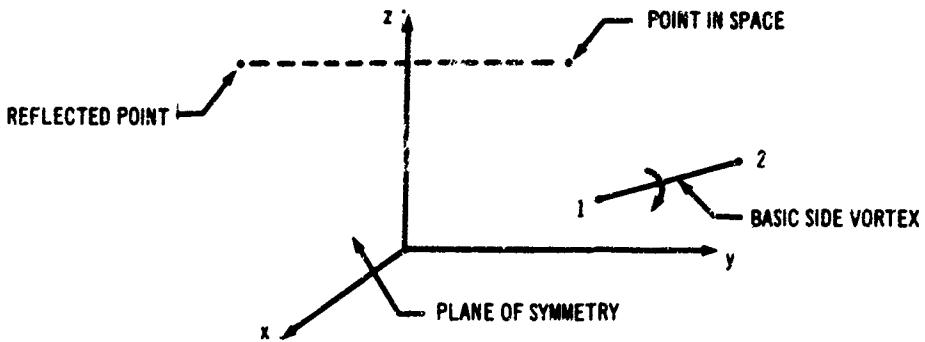


Figure 94. Subroutine VOREQ—Line Vortex in a Symmetric Flow Field.

Thus, the signs of  $y$  and  $B$  are changed, and the velocity components due to the reflected vortex are computed using the above equations.

For symmetric flow, the velocity components are

$$V_x = V_{x(\text{basic})} + V_{x(\text{reflected})}$$

$$V_y = V_{y(\text{basic})} + V_{y(\text{reflected})} \quad (252)$$

$$V_z = V_{z(\text{basic})} + V_{z(\text{reflected})}$$

USAGE:

CALL VOREQ (NSYM, X1, Y1, Z1, X2, Y2, Z2, X,  
Y, Z, VX, VY, VZ)

Input: NSYM = 1, unsymmetric flow

2, symmetric flow

$X_1 \left. \begin{matrix} \\ Y_1 \\ Z_1 \end{matrix} \right\} = \text{coordinates of vortex end point 1}$   
 $(x_1, y_1, z_1)$

$X_2 \left. \begin{matrix} \\ Y_2 \\ Z_2 \end{matrix} \right\} = \text{coordinates of vortex end point 2}$   
 $(x_2, y_2, z_2)$

$X \left. \begin{matrix} \\ Y \\ Z \end{matrix} \right\} = \text{coordinates of the point in space}$   
 $(x, y, z)$

**Output:**  $\left. \begin{matrix} V_x \\ V_y \\ V_z \end{matrix} \right\}$  = velocity components ( $V_x, V_y, V_z$ )

**SUBPROGRAMS  
CALLED:** None

**ERROR RETURNS:** None

**RESTRICTIONS:** None

**SUBJECT:****ASCENT Subroutine WRTETP****PURPOSE:**

To write a matrix on a data file

**METHOD:**

The writing is done with a format consistent with the Boeing-developed matrix routines.

**USAGE:**

DIMENSION A(JR, N), B(12)

CALL WRTETP (A(1, 1), JR, NAME, M, N, B, NF,  
NMT, NTAPE, IRROR)

**Input:** A = the matrix to be written on  
the data file. If this is not A(1, 1),  
the designated matrix will go on the  
data file.

JR = maximum matrix size. This also  
must be the row DIMENSION state-  
ment entry for A.

NAME = integer name of the matrix

M = number of rows in the matrix

N = number of columns in the matrix

B = scratch array for use by WRTETP

NF = number of end-of-file marks to be  
passed before writing starts. If 0,  
no file spacing takes place

NMT = number of matrices to be passed  
before writing occurs. If 0, no  
matrix spacing takes place

NTAPE = number of the data file containing  
the matrix

**Output:** IRROR = error code

**SUBPROGRAMS****CALLED:**

None

**ERROR RETURNS:****IRROR** = 0, if successful write

= 1, if file spacing error

= 2, if matrix spacing is negative

= 3, if matrix spacing error occurs

**RESTRICTIONS:**

None

C. Boundary-layer program subroutine descriptions. — The subroutines listed alphabetically below are discussed in this section.

Subroutine Index

<u>Subroutine</u>	<u>Page</u>	<u>Subroutine</u>	<u>Page</u>
DED	311	INTERP	316
DEDIS	311	MAIN	317
DEI	311	MEAN	319
DES	311		

**SUBJECT:**

**FORTRAN IV Subroutines DED, DEDIS, DEI, DES**

**PURPOSE:**

To solve a set of N first-order ordinary differential equations by the Adams-Moulton predictor-corrector method

**METHOD:**

The differential equations must be in the form

$$y'_1 = f_1(t, y_1, y_2, \dots, y_n)$$

$$y'_2 = f_2(t, y_1, y_2, \dots, y_n)$$

----- (253)

$$y'_1 = f_1(t, y_1, y_2, \dots, y_1, \dots, y_n)$$

$$y'_n = f_n(t, y_1, y_2, \dots, y_n)$$

t is the independent variable and  $y_1, y_2, \dots, y_n$  are the dependent variables that must be determined for a given range of t.

A description of this method follows:

1. The initial conditions for  $t, y_1, y_2, \dots, y_n$  must be given as well as an initial independent variable increment h. Backward integration may be performed by using a negative h.
2. Each differential equation requires four points before the predictor-corrector equations may be applied. The first three integrations are performed by the Runge-Kutta method, and the initial conditions give the required starting points for the predictor-corrector formulas. Each time the step size is altered, it is necessary to obtain new starting points using Runge-Kutta integration.
3. Once four points are determined, the integration is performed from point n to  $n + 1$  along the curve using Equation (254) to predict and Equation (255) to correct.

$$y_{i,n+1}^{(p)} = y_{i,n} + \frac{h}{24} (55 f_{i,n} - 59 f_{i,n-1} + 37 f_{i,n-2} - 9 f_{i,n-3}) \quad (254)$$

$$y_{i,n+1}^{(c)} = y_{i,n} + \frac{h}{24} (9 f_{i,n+1}^{(p)} + 19 f_{i,n} - 5 f_{i,n-1} + f_{i,n-2}) \quad (255)$$

where (p) and (c) represent predicted and corrected values, respectively. If the

corrected value is satisfied so that it is not required to change  $h$ , the derivatives  $y'_{1,n+1}$  are calculated from the differential equations.

USAGE:

More than one system of differential equations may be solved in a given program. Let  $N$  be the maximum number of differential equations required to be solved in a program. The following must be considered in using the routines.

1. An array of  $12N + 9$  storage cells must be reserved in a dimension statement. This block will be referred to as  $T$ .
2. Dealing with a particular system of differential equations, the number of equations must be stored in two locations, referred to as  $NUM$  and  $NUM1$ .
3. An equivalence statement giving  $T(1)$  and  $NUM1$  the same working location must be written.
4. Parameters input to the differential equation routine are listed below:

$\beta$  = fractional part to decrease the interval  $h$  if it needs to be reduced. Normally,  $\beta = 0.5$ ; however, it must be in the range  $0 < \beta < 1$ .

$A$  = number used in determining the relative error between the predicted and corrected values. Normally,  $A = 1$ .

$h_{\min}$  = minimum increment of the independent variable. This value must not be negative.

$h_{\max}$  = maximum allowable increment of the independent variable. This value must not be negative.

$M$  = ratio between upper bound and lower bound relative error. Normally, 1000.

$E$  = allowable relative error between the predicted and corrected values. A representative value is  $10^{-6}$ .

5. The following quantities must be initialized with the indicated quantities prior to entering the differential equation-solving routines.

NUM, NUM1	= number of differential equations that must be solved
T(12*NUM)	= initial value of 1 <sup>st</sup> dependent variable
T(12*NUM-1)	= initial value of 2 <sup>nd</sup> variable

T(12*NUM-NUM+1)	= initial value of last dependent variable
T(12*NUM+1)	= initial increment of independent variable
T(12*NUM+2)	= initial value of independent variable
T(12*NUM+4)	= $\beta$
T(12*NUM+5)	= $h_{\min}$
T(12*NUM+6)	= $h_{\max}$
T(12*NUM+7)	= A
T(12*NUM+8)	= M
T(12*NUM+9)	= E

6. A block in the program or a subprogram must be made to calculate the derivatives. These must be calculated in terms of T-block locations, since the differential equation subprogram links to this block to compute the derivatives when required. The coding of the derivative calculations may be simplified by equivalence statements, if the number of differential equations is invariant.

T(11*NUM)	= first derivative $y'_1$
T(11*NUM-1)	= second derivative $y'_2$
T(11*NUM-NUM+1)	= last derivative $y'_n$

7. There are three linkages to the differential equation-solving routine, DEDIS, by CALL statements. They are given the names DES, DEI, and DED.
- a. DES used once at the beginning to prepare the routine for integration of a given set of differential equations. The CALL statement is CALL DES(NUM1, IND, NTEST).
  - b. DEI used to integrate all  $y_i$  from  $t_{initial}$  to  $t_{final}$ . The CALL statement is CALL DEI(NUM, IND, NTEST).
  - c. DED the CALL statement is CALL DED(NARG).

8. The following fixed-point arguments are required for the CALL statements:

NUM } = number of equations in the system  
NUM1 }

IND = 0, variable interval using Adams-Moulton predictor-corrector method.

NTEST = used in the differential equation program to obtain the correct linkage to the differential equation routine. Its value is set in the differential equation routine. It must be the third argument of DES and the argument of the IF statement, as indicated in the example below.

NARG = calculated in the differential equation subprogram and tested in the calling sequence to determine whether an integration step is completed.

#### Example Calling Sequence

```

CALL DES(NUM1, IND, NTEST)
7 IF(NTEST) 15, 15, 11
11 CALL DEI(NUM, IND, NTEST)
15 (Derivative Calculation)
CALL DED(NARG)
IF(NARG. EQ. 0) GOTO 7

```

SUBPROGRAMS  
CALLED:

None

**ERROR RETURNS:** If an error is detected, a comment is printed and exit is made to the monitor. Checks are made for the following errors:

<u>Error</u>	<u>Error Comment</u>
1. Number of differential equations $N = 0$	Differential equation subroutine DES input error $N = 0$
2. $N > 4096$	Differential equation subroutine DES input error $N = 4096$
3. Step size $H = 0$	Differential equation subroutine DES input error $H = 0$
4. Integration mode indicator $\neq 0, 1, \text{ or } 2$	Differential equation subroutine DES input error IND
5. Predictor-corrector tolerance $E = 0$	Differential equation subroutine DES input error $E = 0$

**RESTRICTIONS:** None

**SUBJECT:** FORTRAN IV Subroutine INTERP

**PURPOSE:** To interpolate for  $y$  given a value of  $x$ , from an array of points  $(x_1, y_1), (x_2, y_2), \dots (x_l, y_l)$

**METHOD:** Assume  $x_{n-1} \leq x \leq x_n$ . A second-order Lagrangian interpolation through the points  $x_{n-1}, x_n, x_{n+1}$  is made to obtain curve  $y_1(x)$ . Another second-order Lagrangian interpolation through the points  $x_{n-2}, x_{n-1}, x_n$  is made to obtain  $y_2(x)$ . Then  $y$  is calculated as the weighted mean of the two curves  $y_1(x)$  and  $y_2(x)$  (see Figure 95).

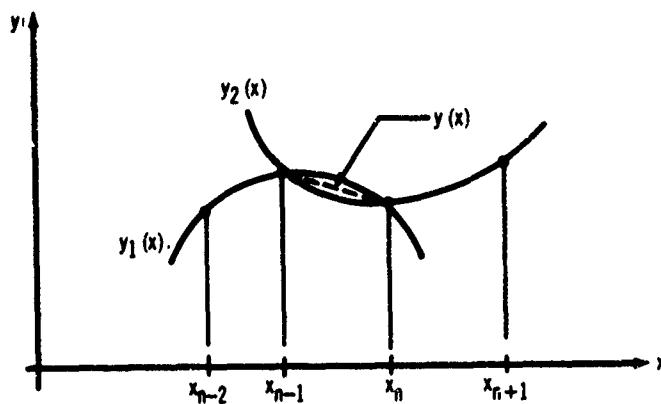


Figure 95. Subroutine INTERP—Lagrangian Interpolation Scheme.

**USAGE:**

```
DIMENSION Y(2000, 7), XX(7)
CALL INTERP (X, Y, IT, XX)
```

Input:     $X$  = value of the independent variable,  $x$   
            $Y$  = array of  $y$  values defining the curves as  
             functions of  $x$   
            $IT$  = number of  $x$  stations

Output:  $XX$  = array of interpolated  $y$  values for the  
             given value of  $x$

**SUBPROGRAMS  
CALLED:** None

**ERROR RETURNS:** None

**RESTRICTIONS:**  $Y(IT, 1)$  contains the values of the  $x$  coordinates and must be monotonically increasing.

SUBJECT:

FORTRAN IV Program MAIN

PURPOSE:

To compute turbulent boundary-layer growth by the simultaneous solution of two first-order ordinary differential equations describing the boundary layer

METHOD:

The following momentum and moment-of-momentum equations are solved simultaneously, along with the skin friction equation, to yield the momentum thickness  $\theta$ , the shape factor  $H$ , and the skin friction  $C_f$ .

#### Momentum

$$\frac{d\theta}{ds} + \theta \left[ \frac{1}{V} \frac{dV}{ds} (2 + H) - K_1 \right] = \frac{C_f}{2} \quad (256)$$

#### Moment of Momentum

$$\begin{aligned} \frac{dH}{ds} = - \frac{1}{V} \frac{dV}{ds} & \left[ \frac{H(H+1)(H^2-1)}{2} \right] + \\ & \frac{H^2-1}{\theta} \left[ H - (H-1) \left( \frac{H_0+1}{H_0-1} \right) I_0 \right] \frac{C_f}{2} \quad (257) \end{aligned}$$

#### Skin Friction

$$\frac{C_f}{2} = 0.123 (10^{-0.678H}) R_\infty^{-0.268} (V\theta)^{-0.268} \quad (258)$$

where  $K_1 = - \frac{1}{V} \cdot \frac{dv}{dy}$

$$\log_{10} H_0 = 0.599 - 0.198 \cdot \log_{10} (R_\infty \cdot V\theta) +$$

$$0.0189 \cdot [\log_{10} (R_\infty \cdot V\theta)]^2$$

$$I_0 = \frac{H_0}{H_0+1} \left[ 1 + .0378 \frac{(52.9 \log_{10} H_0 - 4.18)^{1/2}}{(H_0^2 - 1)} \right]$$

**USAGE:**      Input: Refer to Section 3.3.2.  
                  Output:  $Y(I, 1) = s$ , an array of arc lengths along the streamline  
 $Y(I, 2) = C_f$ , skin friction coefficient at  $s$   
 $Y(I, 3) = \theta$ , momentum thickness at  $s$   
 $Y(I, 4) = H$ , shape factor at  $s$   
 $Y(I, 5) = V$ , velocity at  $s$   
 $Y(I, 6) = \frac{dV}{ds}$ , velocity gradient at  $s$   
 $Y(I, 7) = \frac{dV_t}{ds_t}$ , gradient of the normal component of velocity normal to the streamline at  $s$

**SUBPROGRAMS CALLED:** MEAN  
                          DES  
                          DEI  
                          DED  
                          INTERP

**ERROR RETURNS:** None

**RESTRICTIONS:** The values of  $s$  must be monotonically decreasing.

**SUBJECT:** FORT<sub>R</sub>TRAN IV Subroutine MEAN

**PURPOSE:** To smooth a set of data points defining a single-valued function of one variable

**METHOD:** A sliding operation that deals with four points at a time ( $(x_i, y_i)$ ,  $i = 1, 2, 3, 4$ ) is employed. Points whose ordinates may be changed are the interior two points at each setting of the operation. The abscissas of the input points are not changed. A straight line is drawn from  $(x_1, y_1)$  to  $(x_3, y_3)$  and another straight line from  $(x_2, y_2)$  to  $(x_4, y_4)$ . If the abscissa of the intersection of these two lines lies in the interval  $x_2$  to  $x_3$ , then the points are left as they are, and the operation slides over one point and begins anew. If the lines are parallel or if the abscissa of the intersection is outside the interval  $x_2$  to  $x_3$ , then new ordinates  $y_2$  and  $y_3$  are calculated. An ordinate on the first line is evaluated at the abscissa  $x_2$ ; similarly, an ordinate on the second line is evaluated at the abscissa  $x_3$ . The new values  $y_2$  and  $y_3$  are the mean values of the original ordinates and new ordinates so calculated.

This operation slides along the set of points until all interior points have been subjected to smoothing. Five such passes are made over the curve at each call of MEAN.

**USAGE:** CALL MEAN (X, Y, N)

Input: X = array of abscissas; a strictly monotone sequence

Y = array of ordinates to be smoothed

N = number of points

Output: Y = smoothed ordinates (the first and last points are not subjected to smoothing)

**SUBPROGRAMS CALLED:** None

**ERROR RETURNS:** None

**RESTRICTIONS:** The abscissas must be strictly monotonic; thus vertical slopes are excluded.

**UNCLASSIFIED**

Security Classification

DOCUMENT CONTROL DATA - R & D	
(Security classification of this body of abstract and indexing annotation must be entered when the overall report is classified)	
1. ORIGINATING ACTIVITY (Corporate name)	
<b>THE BOEING COMPANY</b> Commercial Airplane Division P.O. Box 707, Renton, Washington 98055	
2a. REPORT SECURITY CLASSIFICATION <b>Unclassified</b>	
2b. GROUP	
3. REPORT TITLE <b>A General Method for Determining the Aerodynamic Characteristics of Fan-in-Wing Configurations—Volume II, Computer Program Description</b>	
4. DESCRIPTIVE NOTES (Type of report and inclusive dates) <b>Final Report</b>	
5. AUTHORIZE (First name, middle initial, last name) <b>Gary R. Hink</b> <b>Richard F. Gilbert</b>	
6. REPORT DATE <b>December 1967</b>	
7a. TOTAL NO. OF PAGES <b>133</b>	
7b. NO. OF REFS <b>4</b>	
8. CONTRACT OR GRANT NO. <b>DA 44-177-AMC-323(F)</b>	
9. PROJECT NO.  <b>c.</b> <b>Task 1F125901A14234</b>	
10. DISTRIBUTION STATEMENT  <b>This document has been approved for public release and sale; its distribution is unlimited.</b>	
11. SUPPLEMENTARY NOTES <b>None</b>	
12. SPONSORING MILITARY ACTIVITY <b>Department of the Army</b> <b>U.S. Army Aviation Materiel Laboratories</b> <b>Fort Eustis, Virginia 23604</b>	
13. ABSTRACT  <b>This report describes a digital computer program developed to study the aerodynamic characteristics of fan-in-wing configurations. The program is written in the FORTRAN IV and ASCENT languages for the Control Data Corporation 6000-series digital computers. Three basic packages are provided by the program: a geometry package produces a detailed description of the configuration, an aerodynamic package provides a theoretical solution for the potential flow about the configuration, and a boundary-layer package furnishes the boundary-layer characteristics on the wing surface. The report provides a description of the program, flow charts and segmentation structure diagrams, and input data formats.</b>	

**DD FORM NOV 65 1473**

**UNCLASSIFIED**

Security Classification

**UNCLASSIFIED**

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Potential Flow Incompressible Flow VSTOL VTOL Lifting Wing Computer Program						

**UNCLASSIFIED**

Security Classification

10723-67